MAINTAINING PRIVACY DURING CONTINUOUS MOTION SENSING

BY

JOSHUA PAUL-JOSEPH JUEN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Doctoral Committee:

Associate Professor Nikita Borisov, Chair
Professor Bruce Schatz
Associate Professor Matthew Caesar
Associate Professor Romit Choudhury

ABSTRACT

Mobile devices contain sensors which allow continuous recording of a user's motion allowing the development of activity, fitness and health applications. With varied applications, the motion sensors present new privacy problems which require protection. This dissertation builds on previous work with activity and fitness machine learning techniques demonstrating the ability to predict medical values from motion data using smartphones. We conduct two clinical trials collecting a data set of eighty-eight patients and forty-five hours of monitoring to analyze the privacy implications of releasing motion data. We extract a comprehensive set of statistical features from all available smartphone sensors and evaluate feature selection techniques and machine learning models. We find we can predict user identity, phone identity, speed, FEV1/FVC, and activity from the motion signal.

Designing a privacy protection mechanism for motion data requires a precise understanding of how the signal predicts the sensitive information. We develop algorithms to conduct private feature selection which identifies features useful for prediction. We find that simply blocking all private features significantly reduces the usefulness of the signal for other predictions. We develop a sensitivity estimation framework to calibrate the noise for each private feature requiring an order of magnitude less noise than differential privacy sensitivity. We find adding noise to private features calibrated using the sensitivity estimate is effective at reducing the prediction of five tested target predictions. Our methods hide both user and phone identification while allowing other prediction but cannot hide activity, FEV1/FVC and speed without significantly lowering the accuracy of other predictions. Our methods are still effective when the attacker has prior knowledge of the noise distribution. The methods presented in this dissertation demonstrate the need for privacy in motion data and provide a framework for protecting sensitive user information in motion readings.

To My Family

ACKNOWLEDGMENTS

The work in this dissertation is the culmination of many hours of advice and discussion from my advisor Nikita Borisov. I would like to thank him for his guidance and teaching which have been instrumental for me throughout my graduate career. I would like to thank Bruce Schatz who has been like a second advisor to me. His guidance in machine learning and big data analysis of the cell phone signals was invaluable to this work. His tenacity and persistence to conduct clinical trials collected the data that made this dissertation possible. I would like to thank the other members of my committee: Matthew Caesar and Romit Choudhury for giving advice to improve the dissertation. The work that follows would not be possible without the hard work and guidance from my committee.

Throughout my graduate career, my education has been impacted by many professors and students at the University of Illinois. I would like to thank everyone in the Hatswitch research group Amir Houmansadr, Robin Snader, Qiyan Wang, Sonia Jahid, Giang Nyugen, and Xun Gong. I would like to give a special thanks to Prateek Mittal for his advice and collaboration with all my work related to AS-path prediction and Tor. I would also like to thank Anupam Das and Aaron Johnson who collaborated with me conducting Internet path measurements of Tor. I would like to thank Qian Cheng who has worked tirelessly with me to collect and analyze motion data from cell phone sensors during the clinical trials which formed the data set used in this dissertation. I would like to thank Carl Gunter, Zbigniew Kalbarczyk, and Donna Brown for their guidance as I worked as a teacher's assistant. I give special thanks to Susan Hinrichs who allowed me to assist in the cyber security laboratory. My work would not be possible without the collaboration of graduate students: Michael Rogers, Jeff Green, David Bergman, Dong Jin, Naoki Tanaka, Joseph Sloan and Yitao Liu who collaborated on various projects over the years providing insight and perspective to improve the

research. I would like to thank the staff at the University of Illinois for creating a learning environment conducive to research.

I would like to thank all the professors at Central Michigan University who work tirelessly to prepare students for a career in engineering. My background in basic principles has made my graduate education possible. A special thanks to James Morrison, Koblar Jackson and Albert Peng who encouraged me to pursue a graduate degree. I would like to thank the contributors to the Centralis Scholarship. Your generous donations allowed me to begin this journey.

Everything I have done is only possible due to the love and support from my family. I would like to thank my father and mother Don and Michele Juen who have encouraged and supported me as I pursued my education. Your guidance and teaching prepared me for success in college and in life. A special thanks goes to my mom who stayed home to homeschool me from kindergarten through high school. I also want to thank my sister Bethany who has always encouraged me when times have gotten tough. I would like to thank my girlfriend Noelle for her love and support as I pursued my doctorate. I would like to thank my Lord and Savior Jesus Christ with whom all things are possible. Thank you to all of my close family and friends who have supported and encouraged me throughout the years. I dedicate this dissertation to those who have encouraged, helped, and challenged me and to God above who watches over us all.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Overview

The rise in popularity of mobile devices presents new challenges to protecting the privacy of users. While users are enjoying the many added conveniences mobile devices provide throughout their daily lives, most users are unaware of the new streams of continuous personal information being collected. Mobile fitness devices are being designed to track a user's every movement in order to measure fitness and activity levels. Medical devices are being developed to track health with the goal of allowing medical practitioners to more accurately diagnose and treat chronic disease. Even smartphones contain motion sensors capable of measuring a user's daily life through the use of global positioning satellite (GPS) tracking location, motion sensors tracking actions and of course communication patterns tracking social behavior and browsing history tracking user behavior. While these devices can gather a wide array of personal information about their users, we focus on the ability of mobile devices to monitor a user's motion using internal motion sensors. While the privacy leaks from GPS, browsing history, and social contact have been studied, motion from privacy sensors has gained little attention from privacy research. The widespread adoption of devices containing motion sensors combined with recent interest from commercial companies to monitor health, such as Apple's HealthKit initiative, creates a need to understand information leaks contained in motion sensor data.

Health trackers, fitness devices, and mobile phones contain motion sensors including accelerometers, gyroscopes and magnetometers to measure a user's motion through space. Motion through space is useful for many applications including gaming, fitness tracking, and giving the phone context awareness with new creative applications being continuously developed. Recent work

has demonstrated that motion sensors may leak potentially private information including user identification, device identification, activity recognition, walking pattern identification, and health status tracking. We expand on previous work by conducting numerous health studies finding that data from these sensors can leak medical metrics giving insight into users' private health information. Our preliminary studies motivate the need to educate users to the potential dangers of releasing private information and develop new privacy architectures which ensure private data is protected.

The methods presented in this dissertation can be used to generally protect predictions made from machine learning models trained with motion data; however, we specifically focus on privacy threats which arise while carrying mobile smartphones. While the privacy threats which arise from carrying fitness and medical devices must be understood, most users are naturally more conscientious while carrying a fitness or medical device because it is only worn while the user is either exercising or conducting medical test. Conversely, a phone is typically always carried for communication purposes and is currently not viewed as a threat to leaking the user's identity, activity or health status through motion data. We demonstrate that the sensors in a phone are similar or better than sensors contained in popular fitness and health devices. We develop software to collect sensor readings on Android-based smartphones. We then conduct two clinical trials to demonstrate the viability of collecting both health and activity data using smartphones on a wide range of patients. The results from these tests demonstrate the viability of smartphones to collect sensitive health information useful for health diagnosis from both chronic and healthy subjects using sophisticated machine learning models. Thus, we demonstrate mobile devices leaking sensitive health information which motivates the need to design and develop privacy protection mechanisms.

In order to design privacy mechanisms for motion sensors, we must understand how raw sensor signals can be useful to make predictions about sensitive fitness, health, and demographic information. We assemble the data from two clinical trials with fifty-eight subjects and combine it with two years of continuous collection from thirty volunteers giving a unique data set comprising eighty-eight subjects and over two gigabytes of raw motion data taken from ten unique smartphones. Using this data, we conduct a detailed study using machine learning to predict thirty pieces of private information includ-

ing health metrics, fitness metrics, user identification, phone identification and various demographic information. We design and implement an analysis pipeline which extracts seventy-four statistical features from thirty-one sensor streams from each mobile device. Our pipeline generates 2,294 total sensor features calculated from continuous windows of data. We then train and evaluate machine learning algorithms to predict each sensor target. We find motion data useful to predict phone identification, user identification, walking speed, and chronic obstructive pulmonary disease (COPD) diagnosis motivating the need for privacy protection to give a user control over the release of this information.

Protecting privacy against machine learning analysis requires the development of new analysis frameworks. We begin by studying the ability of standard feature selection routines to identify the set of statistical sensor features which leak private information. We find that traditional feature selection techniques are not designed nor suited for identifying the complete set of private features. We propose three algorithms to return the top set of private features and validate each using machine learning models. We find a tradeoff between computation power and accuracy. We find an algorithm which clusters features using normalized mutual information and raw prediction scores during the first round of a sequential forward search mechanism using a support vector machine/regression correctly classifies most private features while limiting the number of false positives. However, it also takes far more computation than using the normalized mutual information score between the feature vector and target vector which is the most accurate filter method. We use our algorithm to identify the private features for each of our thirty prediction targets.

The utility of any privacy mechanism is limited depending on how severely it degrades legitimate performance. The simplest way to protect motion data is to block the release of all private features. However, we find many prediction targets have overlapping private features. We investigate the ability to introduce sufficient noise to obfuscate a sensitive prediction target while still leaving enough of the signal to be useful by developing methods to estimate the sensitivity of each statistical sensor feature to added noise. We design a framework in Python capable of estimating sensitivity for both specific and generalized machine learning models. For classification, the algorithm estimates the average distance the feature must change to change the clas-

sification output. For regression, the framework estimates the change in prediction output per change in input feature values. Our framework allows careful estimation of the required noise needed to obfuscate prediction.

Our analysis of each stage of the machine learning pipeline allows us to use principles from differential privacy to design obfuscation routines. Using the private features for each target and sensitivity analysis, we can introduce noise into the signal significantly reducing the predictive ability of the machine learning while minimizing the collateral impact on the accuracy of predicting other targets. We test traditional differential privacy noise estimation against our sensitivity routines finding our sensitivity estimates requires on average an order of magnitude less noise to hide the target prediction. We find we can add sufficient noise to reduce classification accuracy of phone identification and user identification without significantly decreasing accuracy of health and fitness metrics. We find that our obfuscation techniques decreases prediction accuracy across all tested machine learning model types for each prediction target. Knowledge of the noise level can help an attacker get better prediction accuracy but the benefit to collateral features is higher than to the protected private feature. Overall, our analysis indicates that regression is more sensitive to changes in noise with classification being less sensitive to input noise. However, more testing on larger data sets using our framework will be required to design a widely deployable system capable of protecting privacy.

We find that motion sensors can leak sensitive information including health metrics, fitness metrics, user identification, and phone identification. We find various demographics more difficult to predict from our data. Our analysis framework allows us to identify private features, estimate the sensitivity of those features for each prediction, and add noise to obfuscate the predictive accuracy to a user configurable threshold. We believe this work motivates the education of users to the dangers of releasing motion sensor information and the presented frameworks can be used to develop better access control capable of selectively protecting sensitive information. Such frameworks will be necessary to give users control over their personal information while allowing motion sensors to be used for the wide variety of creative applications currently being deployed on mobile devices.

1.2 Motivation

Mobile technology has gained widespread acceptance in our society. Mobile phones provide users internet connectivity from virtually anywhere allowing open access to information regardless of location. Location-based services provide instant directions to help users by suggesting places to eat, shop, relax or work. Fitness devices are gaining popularity to improve health by allowing a user to track various fitness metrics including step counts, distances walked, caloric expenditure and other exercise measurements. Dedicated health devices are being developed to track various conditions ranging from general measures of health such as heart rate, blood pressure, oxygen saturation to specific disease diagnosis such as cardiopulmonary function or asthma inhaler use. From phones to dedicated devices, we are seeing vast usage of mobile platforms to monitor and assess health and fitness in users. All these devices present platforms capable of continuously collecting data about their users from a variety of sensors. While this presents unprecedented utility for analysis in order to improve health tracking, it presents the ability for side-channel attacks against a user's privacy. Therefore, the data which is returned from these devices must be carefully analyzed in order to determine the privacy implications to the users.

While the development of fitness trackers and health devices has taken place in parallel to mobile phones, the phone manufacturers are currently exploring the capabilities of the phones to mimic the functionality of dedicated devices. Health apps are gaining popularity in the smartphone marketplaces with hundreds of new apps appearing in both the Apple and Google Play stores. Both Apple and Samsung are including health fitness apps in the base installs of the operating systems making fitness trackers part of the core phone functionality capable of tracking movement, steps taken, and caloric expenditure. New phones are also including what have traditional been medical measures including heart rate, blood oxygen level, and respiratory rate among others. Apple's Health Kit initiative is designed to make continuously collected sensor data available in medical records and the Research Kit initiative has been developed to provide all the collected information to clinical research teams. These initiatives are moving continuous monitoring to standard practice on modern smartphones and creating easy ways for researchers and other apps to access the data.

Users have been extremely tolerant of giving vast amounts of personal information to phone apps in order to make their lives easier. While alarming, this has led to a relatively relaxed privacy policies on most smartphones in use today. Unfortunately, the access to information on current mobile devices opens users to unprecedented exposure of all collected private information. The ability of mobile devices combined with a history of inadequate security creates the possibility of a user's entire life being continuously tracked, recorded, and analyzed. While the new medical devices will undoubtedly be carefully protected through strict access control, the phone, which contains almost identical motion sensors, will probably continue to be considered low risk when releasing a user's motion information. It is unclear how successful fitness and medical monitoring will be; however, including health related measurements continually edges closer to being classified as health information which is strictly protected under the Health Insurance Portability and Accountability Act (HIPAA). While the necessary privacy requirements to protect continuously collected data from mobile devices is currently unclear, the ability of the phones to match dedicated health devices will probably warrant better privacy protections. Thus, it is critically important to determine the capability of the smartphones to continuously collect sensitive health information. Whether classified as protected health information or not, such data collection introduces the need for higher security and privacy on mobile devices. At the very least, users must be educated on the potential risks of providing their data to both researchers and other apps. In order to classify the risks, privacy research must understand and quantify what types of inferences can be conducted by releasing information from the various sensors in the devices.

This dissertation hopes to parallel previous research in side-channel attacks on GPS readings which have led to better understanding for the need to control access to the GPS. Unfortunately, motion sensors, specifically the accelerometer, magnetometer and gyroscope, are still easily available to an adversary due to minimal protection by access control. Motion sensors are even available through the web browser allowing an attacker to take readings when a user visits a malicious site. Such policy is dangerous if there is private information contained in the raw readings themselves. Privacy research must therefore investigate the possible inferences and design systems to protect sensitive user information. Such systems are necessary to ensure a user's

Figure 1.1: Various Possible Privacy/Sensitivity Relationships for (G) Gender and (ID) User Identification

privacy and maintain public confidence in mobile technology.

Access to motion sensors provides access to a wide variety of motion features which can be measured as the user carries the device. Protecting privacy against inferences requires a thorough understanding of what specific subset of motion features are most important for classifying private information. Specifically we wish to find the subset of primary features contained in the motion sensor data that predicts each private characteristic. It is not only important to understand what primary features are most useful to predict a characteristic but also how sensitive the prediction is to variations in those features in order to design privacy systems. The sensitivity is defined as the amount of change in a given input to change the output value. As an example, assume we are analyzing the sensitivity to two classifiers, user identification (ID) and gender (G). In general, we can envision three outputs to our sensitivity analysis as shown in Figure 1.1. If the primary features that identify the user are a mutually disjoint set from the set of primary features that identify gender, then it should be possible to add noise to the primary features indicating each characteristic and protect privacy in both without destroying the other classifier. If the set of two features contains intersecting primary features, but the sensitivity of features predicting gender is less than the sensitivity of the features predicting user ID, then it should be possible to add an appropriate amount of noise to obfuscate ID while leaving enough information to predict gender. Finally, it may be the case that the features are intersected and the sensitivity is similar in which case the classifiers are not separable and it would not be possible to hide one without hiding the other. Determining what characteristics can be inferred from motion sensors, what primary features in the motion sensors indicate a target characteristic

and how sensitive the predictions are to changes in the motion sensor features is necessary to design privacy systems for motion sensor data. Once determined, privacy systems may be developed to introduce various types and levels of noise into the signal thereby hiding specific characteristics in the data.

1.3 Dissertation Outline

It is the purpose of this dissertation to establish that raw motion data from the accelerometer, magnetometer, and gyroscope in smartphones contain side-channel information warranting careful protection. Chapter 2 will provide important background in mobile sensing, devices and target characteristics which have been detected in motion data. Many studies have been conducted on dedicated medical and fitness devices but fewer have looked at a smartphone's ability to track health. Thus, the dissertation will first establish that smartphones contain similar if not better capabilities than modern fitness and medical devices. We will build on previous research in using machine learning to infer motion characteristics from accelerometers and develop novel work establishing the ability of motion sensors to detect critical health measures in the readings. This will require conducting multiple clinical trials in order to collect data from patients during natural walking. The clinical trials will collect a unique data set of patient data which will be used for the analysis in the remainder of the dissertation. The validation of smartphone sensors and clinical trials are presented in Chapter 3.

The clinical trials provide a rich data set to analyze in order to establish the threat of side-channel information in motion data. Figure 1.2 outlines the remaining analysis for the dissertation. We will choose a wide variety of target characteristics to infer from the motion data which will be determined from a combination of previous work and novel characteristics determined from the clinical trials. Chapter 4 will outline what sensor streams are available from testing and what statistical features can be extracted for machine learning. Chapter 5 will determine the set of extracted input features which are most predictive of each target characteristics. We will then build sophisticated machine learning models to predict each characteristic in Chapter 6 and determine which characteristics can accurately be inferred from the

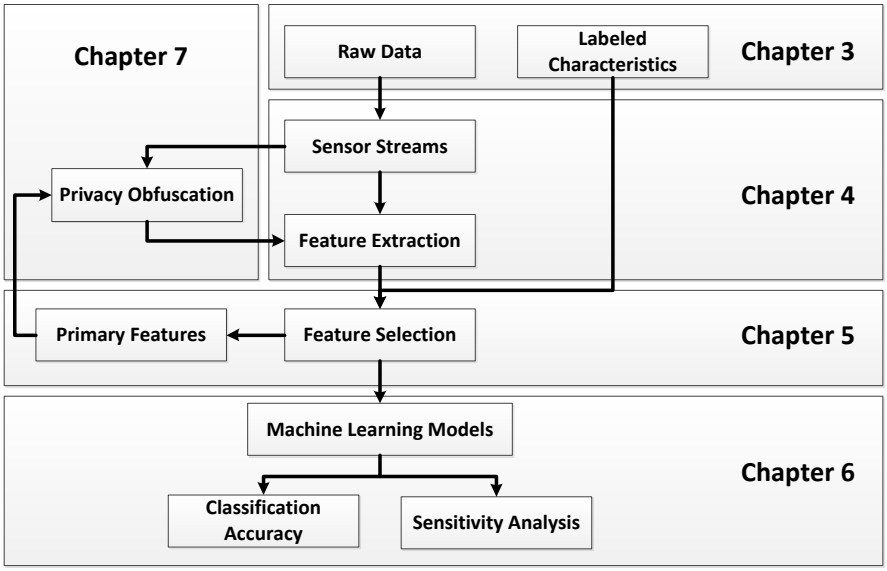

Figure 1.2: Dissertation Outline

motion sensors. We will also evaluate the sensitivity of each inference to the primary features giving an ordering of which target characteristics are easy to predict and which are more subtle in the data.

Once the sensitivities of the critical characteristics are identified, privacy preserving techniques will be developed to hide side-channel information. We will investigate the use of fine-tuned techniques based on differential privacy which attempt to hide specific side-channel information without significantly hampering the usefulness of the sensors for legitimate prediction in Chapter 7. Such systems will undoubtedly cause degradation in the signal quality which must be quantified. The analysis will determine the possibility of designing systems capable of hiding specific inferable characteristics in the signal while still maintaining use in application which are less sensitive to noise. This will also produce a list of inferable characteristics and determine the difficult in obfuscating each characteristic. Finally, we will discuss conclusions and future work in Chapter 8 outlining how the work presented in the dissertation can be used to design privacy obfuscation techniques applicable to the signal in real time.

It is currently unclear how privacy will be handled in a world filled with

mobile devices. Continuous collection of motion data opens users to unprecedented real-time tracking of their behavior. Understanding what characteristics can be inferred and how noise can be injected to hide each piece of information will allow systems to be designed with fine-tunable access control. This will allow users both better understanding of their privacy options with the added benefit of allowing them to obfuscate if necessary. It is the goal of this dissertation that this analysis will both promote awareness of the privacy issues and provide a solution to allow these systems to operate without fear of privacy loss of their users.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Motion Sensors

Modern electronics use various sensors to record the motion of the devices. The most popular sensors in use are accelerometers which measure the relative acceleration of the device, gyroscopes which measure rotation, and magnetometers which measure the magnetic field strength around the device. These sensors combined with some signal processing allow the device to determine the direction of gravity or the downward vector and relative orientation in the world coordinates (in relation to true north) with some level of accuracy.

2.1.1 Magnetic Compass

Measuring magnetic fields in a single dimension is fairly trivial by measuring the Hall effect [1]. When a current is flowing across a conductor, a magnetic field will cause a charge buildup perpendicular to the magnetic field and transvers to the flow of electric current as seen in Figure 2.1. By measuring the voltage of the Hall effect, the relative strength of the perpendicular magnetic field can be determined. While straightforward in one dimension, measuring the relative magnetic field in three dimensions is more challenging. The most obvious solution would be to place three Hall sensors perpendicular to one another; however, minor errors in placement cause large errors in the final measurement. The solution as implemented in modern devices is shown in Figure 2.2 [2]. All three directions are measured in one plane thereby eliminating alignment errors. To accomplish this, a magnetic concentrator is placed on the X plane which bends the X and Y magnetic fields downward into the Hall sensors. The Hall sensors then measure the combinations of

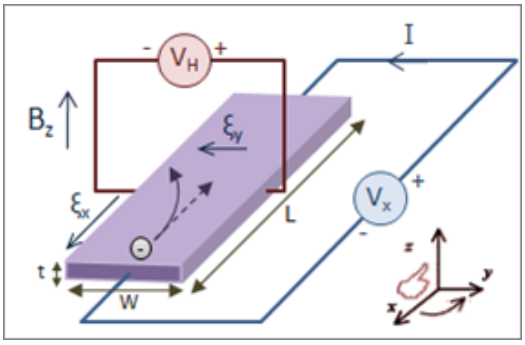

Figure 2.1: The Hall Effect

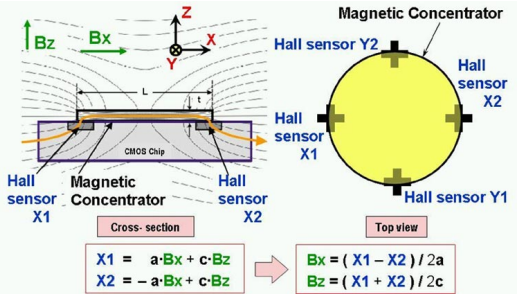

Figure 2.2: Basic Theory of MEMS Compass

magnetic fields strengths. Using two Hall sensors on either side of the concentrator, each sensor measure $X1 = a*Bx+x*Bz$ and $X2 = -a*Bx+c*Bz$ where Bx and Bz are the relative magnetic field strengths in the x and z directions and a and c are constants defined by the physical parameters of the Hall sensors. Then to get the relative magnetic field strength the device simply calculates $Bx = (X1 - X2)/2a$ and $Bz = (X1 + X2)/2c$. Obviously, the By direction can be measured by using a similar set of equations with the two Y hall sensors.

2.1.2 Acceleration

Most modern devices measure relative acceleration using micro electro-mechanical systems (MEMS) accelerometers [3]. Accelerometers measure the relative acceleration of the sensor in a single direction. Accelerometer sensors are normally modelled as a mass on the end of a spring. As the sensor is accelerated, force acts on the mass causing a change in length of the spring. By measuring the length change, the relative acceleration can be determined. Figure 2.3 illustrates the basic outline of a MEMS accelerometer. A moveable plate is attached to a spring with a constant k_s. When an acceleration is applied parallel to the spring, a displacement occurs. The relative capacitance $C1$ and $C2$ change as the plate moves. The chip measures this capacitance and determines the appropriate acceleration. In modern chips, three of these systems are fabricated orthogonal to each other allowing the relative acceleration to be measured in three directions.

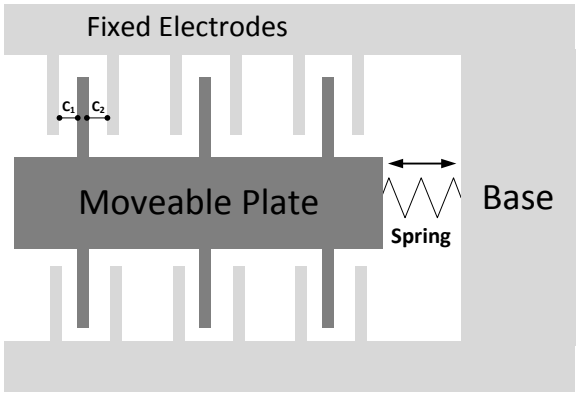

Figure 2.3: Basic Outline of a MEMS Accelerometer

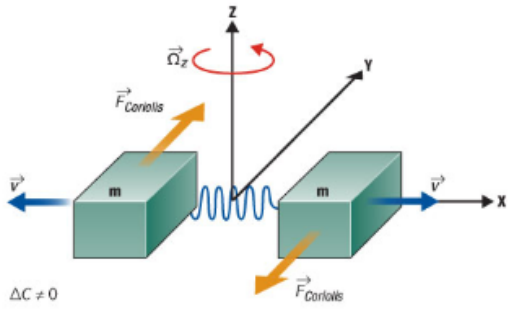

Figure 2.4: Basic Theory of MEMS Gyroscope

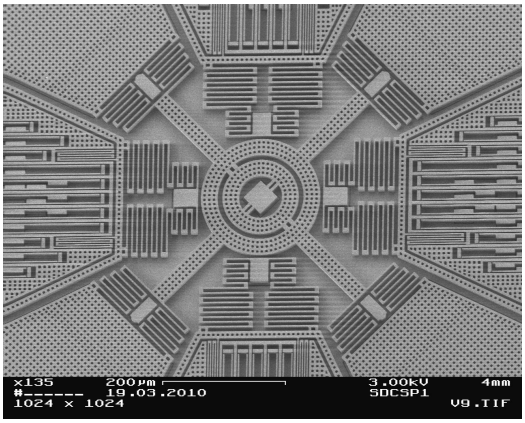

Figure 2.5: X-Ray of MEMS Gyroscope

2.1.3 Rotation

Recently MEMS gyroscopes have gained popularity to increase the precision of measuring device rotation especially in gaming applications. MEMS gyroscopes measure device rotation by measuring the force due to the Coriolis effect [4]. Figures 2.4 and 2.5 demonstrate the theory behind a MEMS gyroscope. Two masses are put into oscillation in opposition to each other along the X axes. When a rotation is applied around the Z axis, the Coriolis effect causes a force in opposite directions along the Y axis. By measuring the difference of the two forces, the amount of rotation can be determined independently of the acceleration in the Y direction.

Thus, modern devices can measure acceleration, rotational velocity and approximate magnetic north; however, there are still significant challenges

to obtaining precise measurements of motion. For example, the accelerometer gives relative acceleration not velocity. A slight error in determining the initial velocity of the phone when estimating displacement using the accelerometer quickly accumulates compounding errors making the measurement incorrect. Gyroscopes are notorious for exhibiting drifting error making it difficult to obtain true stationary rotational velocity without using either the compass or accelerometer. The compass while accurate in a clean environment, must operate in close proximity to speakers and transmitters which generate magnetic interference. Thus, each sensor has significant challenges to overcome when taking measurements. It is also important to note that device identification is made possible by measuring minute difference in sensor fabrication. For example, the measured capacitance on the accelerometer as shown in Figure 2.3 depends on the distance of the stationary electrodes to the electrodes on the movable plate. While MEMS fabrication is fairly precise, extremely small variation in the fin placement between devices can yield detectable differences in the measurement of acceleration [5]. Both MEMS gyroscopes and MEMS compasses can also exhibit minor differences in readings due to variation introduced during fabrication. Detecting these differences will become important when analyzing the ability of identifying the device through this manufacturing "fingerprint."

2.2 The Rise of Dedicated Health and Fitness Monitors

2.2.1 Fitness Monitors

Fitness monitors have recently gained commercial attention. The Fitbit Flex, Jawbone Up and Nike Fuel Tracker are currently the most popular dedicated fitness monitors. These devices track steps taken, raw motion, and caloric estimates for their users. Marketed as devices suitable for entertainment purposes only, the devices still have the capability to provide continuous streams of spatio-temporal motion but limit access to the raw data to their parent companies. While it is currently unclear how much data is being collected, liberal terms of use agreements often allow the companies to collect and analyze a user's spatio-temporal motion without further consent. This makes understanding the utility of this data critical to determine if stricter

permissions should be mandated.

Recently, Fitbit has launched the Fitabase initiative to encourage the use of Fitbit devices in clinical research. Many medical researchers are gaining interest in the accuracy of fitness trackers since they are gaining popular use presenting a potential platform for clinical research without requiring custom, often expensive, equipment. Recently, Lee et al. investigated the accuracy of popular fitness devices finding them to be promising to estimate caloric expenditure [6]. Takacs et al. validated the step counts measured by Fitbits during treadmill walking [7]. These studies are promising for the commercial fitness devices but severely limited in scope. The caloric trial was limited to a short trial within the clinic and the step count trial was limited to treadmill walking which differs significantly to natural walking. This dissertation will contribute an accuracy assessment of popular fitness devices in Chapter 3. Regardless of current accuracy, the motivation for fitness devices to be used in clinical settings will likely lead to improvements of the prediction routines. Thus, protecting the user's data from privacy attacks will most likely be necessary as the device's accuracy continue to improve.

2.2.2 Health Monitors

Health monitors have been a popular topic of research in the custom sensor community over the past ten years. Systems have been proposed and tested which measure activity and gait characteristics accurately using custom hardware often containing motion sensors (accelerometers and gyroscopes). Activity recognition in particular is an extremely well researched area and identifying is perhaps the most frequent activity identified [8, 9, 10, 11, 12, 13, 14]. These studies demonstrate the interest in the medical community to use dedicated medical quality monitors to improve patient care; however, they fail to consider the privacy implications of the recorded motion data.

Clinical researchers have conducted a number of studies measuring patients using medically verified accelerometers. Steele et al. performed an early experiment to classify physical activity using a medical accelerometer strapped around the patient's chest [15]. Moe-Nilssen et al. provided medical validation of using professional medical accelerometers to measure

gait cycle characteristics [16]. In 2006, Pitta et al. added a motion sensor (accelerometer) along with the more accepted medical questionnaires to assess daily physical activity among chronic obstructive pulmonary disease (COPD) patients [17]. In 2013, Rabinovich et al. validated the use of medical accelerometers to monitor patients during their daily life at home [18]. Van Remoortel et al. tested and medically validated six popular activity monitors which are now considered acceptable for use in medical trials [19]. Rabinovich et al. expanded the use of sensor systems by deploying activity monitors to COPD patients to use in their homes [18]. Recently, most studies have adopted the Actigraph GT3X as the standard medical accelerometer to use in clinical trials. The Actigraph give researchers access to the raw accelerometer data opening it to the possibility of side-channel privacy leaks and motivating the need to understand what side-channel information can be leaked from motion sensor data.

Systems capable of performing automated diagnosis of patients are currently being developed due to strong motivation to improve the quality of care while lowering cost in the healthcare industry. In particular, measuring activity and gait speed are important to consider since it has been recognized that a decrease in mobility and motion, especially when walking, is a strong measure of a patient's health when they have both COPD and congestive heart failure (CHF) [20, 21, 17, 22]. All these studies indicate that gait speed and cadence are valuable indicators for health especially in COPD and CHF. A slower cadence indicates poor health useful as an indicator for physicians. As diseases with millions of patients, these two diseases have become the topic of widespread study due to the high volume of hospital readmissions. Since hospital readmissions add significant cost to the healthcare industry, there is a strong desire for automated methods to continually allow diagnosis of patients when they are away from the hospital by monitoring the mobility of the patient while at home. Such systems require strong privacy analysis to protect sensitive side-channel information during continuous data collection. While initial research has not considered the privacy implications of continuous motion monitoring, the final commercial system will need to be designed to alleviate privacy concerns. This requires an investigation of potential privacy leaks in order to design secure systems and motivates the need for the analysis in this dissertation.

2.3 Collecting Spatial Information from Smartphones

Ubiquitous adoption of smartphones to the general public is now a reality. As volume increases and prices decrease, sophisticated smartphones continue to gain market share. Smartphones contain the ability to run moderately sophisticated programs which can interpret various sensor readings including global positioning systems, tri-axial accelerometers, gyroscopes, magnetometers and others. New sensor pipelines promise energy efficient monitoring addressing concerns of battery life. These sensors combined with the computational power of smartphones provide an ideal platform to deploy widespread monitoring systems to the general population. This opens a new opportunity for medical and health measurement and a new potential threat to a user's mobile security. The phone's widespread adoption combined with the ability to adaptively preprocess data gives it an advantage over dedicated medical and fitness devices which are currently gaining popularity; however, phones suffer from questions regarding sensor accuracy. The trials referenced in Section 2.2 have demonstrated the ability of expensive medical grade accelerometers to infer valuable health statistics for clinicians. Thus, a careful investigation must be conducted into the ability of smartphones to infer health information especially compared to dedicated medical devices.

While the analytic techniques used to design private systems in this dissertation can easily be applied to both mobile phones and dedicated devices, we consider the threat from smartphones greater since they are more widely adopted and allow an adversary to passively collect sensor information without the user's knowledge. While privacy is important during data collection using dedicated fitness and medical devices, it is likely the patients are aware they are being monitored while carrying a dedicated device. However, if such measurements are possible using a mere smartphone, it would be possible to leak this health information by giving access to the smartphone sensors to downloaded apps. In contrast, dedicated devices are often only worn during monitoring allowing the user greater control over the release of their motion data.

For phones to be a threat to a user's privacy, they must be shown to have high accuracy similar to dedicated devices. We will demonstrate in Chapter 3 that cell phones contain the same sensor chips as many popular dedicated health and fitness devices. Unfortunately, fundamental design choices in a

phone's firmware can produce difficulties in obtaining a cleanly sampled signal. Phones return sensor readings at "best effort" which can be slowed by other processes running on the phone, but any analysis in the frequency domain requires a steady sampling rate high enough to capture the motions in human movement. Studies on human gait characteristics have established that human walking requires between 3-5 Hz to capture 99% of the signal's power from gait wobble contained in the spectra below 15 Hz [23]. Studies in motion capture have established 15 Hz as ideal to capture hand movements and 7 Hz to capture facial expressions [24, 25]. This requires a sensor capable of providing a clean reading at 30 Hz to obtain the minimum required Nyquist frequency. Fortunately, software post-processing can correct the signal [26, 27]. We will design and develop smartphone software which overcomes phone firmware limitations and collects human motion with similar medical accuracy as the accepted medical accelerometers. We compare our systems using the accelerometers analyzed by Remoortel et al. as our baseline for standard medical accelerometers [19].

One final popular criticism to phones is that unlike dedicated devices which are often affixed to the point of interest on the subject such as the wrist or waist, phones are often carried anywhere on the user's body. Thus, various positions must be considered and corrected in order to infer motion characteristics. Correcting for phone position has been demonstrated using various machine learning techniques [28, 29, 30]. Thus, correcting for various phone placement locations is considered a solved research problem which we choose to not address in this dissertation.

The ability for phones to monitor at similar levels of accuracy as medical monitors naturally leads to a problem with security and privacy. While the low quality of access control on smartphones can (and probably will) be quickly changed to suit best practices, the raw motion sensors are required for many applications ranging from games to simple apps which wake the phone upon shaking. Thus, blocking access to the motion sensors is currently an infeasible solution. Instead, privacy mechanisms must be designed which can intelligently quantify the possibility of side-channel information in order to provide a finer level of access control.

2.4 Inferring Information from Motion Sensors

The popularity of smartphones combined with the richness of the sensors has motivated a large body of work designing and testing various measurement systems. We will now give a broad overview of the various categories.

2.4.1 Activity Recognition

Activity recognition is focused on determining what a phone user is currently doing. The Jigsaw system was built to conduct activity recognition and optimize battery efficiency while sampling from the GPS [31]. It combined accelerometer, microphone and GPS data to attain a user's motion profile. The accelerometer classification used twenty-four selected input variables to various machine learning algorithms. Each algorithm was trained on a specific device placement with the system deciding the proper training to use with a binary decision tree. The output of the classification was whether the user was walking, running, bicycling, in a vehicle, or stationary. The authors report a 95.1% accuracy when testing on a limited trial of college-aged students. In a similar study, Kwapisz et al. trained a neural net classifier to recognize walking, running, jogging, going up stairs, going down stairs and remaining stationary with 91% accuracy on college-aged subjects [32]. In addition to walking, other activities have been classified including driving [33], drunk driving [34], cleaning, eating, meeting, reading, and watching television [8]. Thus, previous work indicates a user's activity information is contained in accelerometer data.

2.4.2 Gait Characteristics

Gait characteristics can often be inferred using motion data collected from users as they walk. This includes walking direction [27], step counting [35], and speed estimation at pre-set walking speeds [14]. Other studies use various machine learning techniques to infer gait speed with users walking on treadmills or at set walking speeds including support vector machines [36], Gaussian process regression [37], linear regression [38], Bayesian linear regression [39], artificial neural networks [40], mechanical models [41, 42] and

simple statistics [43, 44]. These studies confirm the intuition that gait characteristics can be inferred from accelerometer data.

2.4.3 Location Recognition

Location recognition uses the motion sensors in the phone to track the specific location of the user. Accomplice tracks a user's location by using the phone's accelerometer to estimate a list of directional displacements. As a user drives or walks down streets, the system attempts to match the displacements to roadways on a map in order to infer a user's travel. This is threatening to user privacy since most users turn off the GPS to protect their location. Since Accomplice does not require access to the GPS, the user's location information is still inferable [45]. On a smaller scale, multiple dead reckoning systems designed to establish a user's location in a constrained area, such as a building, use accelerometer data to fine-tune their location prediction [46, 47, 48]. Thus, continuous acceleration data can cause leaks to location privacy.

2.4.4 Biometric Identification or Continuous Authentication

Biometric identification uses the motion of the phone to uniquely identify the subject carrying the phone. The idea of using gait as a biometric has been widely studied in computer vision [49, 50, 51]. More recently, accelerometers have been proposed to identify subjects based on their unique gait patterns [52]. Gait-ID identifies subjects with 99% accuracy in limited testing by using a Mexican-hat wavelet transform combined with a support vector machine classifier. A number of papers have recently attempted other machine learning algorithms to also identify users using accelerometer and gyroscopes using various spatial temporal parameters [53, 54, 55]. Besides identifying a specific user, it has also been shown that accelerometers can provide a unique signature for device-specific identification which could allow the device to be fingerprinted across apps or internet sessions [5].

2.4.5 Fitness Metrics

Fitness metrics are quickly gaining popularity to help users track their health. While not considered medical quality, these metrics often can give insight into a user's life and therefore present sensitive information which needs added protection. Note that these characteristics while health related are not medically validated metrics and therefore do not fall under HIPPA's authority to regulate the user's health information. Calorie expenditure is a popular topic for health management with systems being designed to use smartphone motion sensors to measure calories burned during walking and bicycling [56, 57]. Other systems have been designed to measure a user's perceived level of stress through their cell phones [58, 59]. SpiroSmart attempts to use mobile phone sensors to measure a user's lung function [60]. Finally, Rabbi et al. design and demonstrate a system to measure a user's perceived overall wellness in an attempt to correlate this perception with mobility data [61]. While not medically validated, all these studies demonstrate that mobile sensors contain some useful information relating to the overall status of a subject. This information must be protected especially if it can be proven to be correlated with real medical values.

2.4.6 Health Quality Metrics

Limited research has been conducted in utilizing mobile sensor systems to measure physical health features. Yavuz et al. designed a system to monitor patients and predict the likelihood of falling down using smartphones [62]. Rabbi et al. study the ability of various mobile sensor's capability to assess mental and physical well-being in test subjects [61]. The viability of using smartphones to analyze clinical gait was presented by Yang et al. but did not contain extensive trials [43]. A more rigorous investigation of clinical gait monitoring was presented by Nishiguchi et al. [44]. While both works contribute to the idea of using smartphones to monitor gait, neither work assesses the viability of monitoring gait during natural unconstrained walking. We establish the ability of using phones to measure medical quality gait characteristics in Chapter 3.

2.5 Conclusions

Mobile phone sensors show great promise in measuring a variety of information. This dissertation will first develop a platform to measure motion similar to medical grade sensors and validate the platform by measuring gait speed and COPD status in Chapter 3. The dissertation will then focus on protecting privacy related to biometric identification, device identification, activity, fitness metrics, health metrics and demographic information.

CHAPTER 3

CLINICAL TRIALS: COLLECTING MEDICAL READINGS WITH MOBILE DEVICES

In this chapter, we investigate the ability of smartphones to collect high-quality medical data with similar quality and accuracy as validated medical devices during unconstrained walking. Previous work indicates sensitive information including personal identification, device identification, activity recognition, and location are all contained in the mobile phone motion sensor data. Previous work has successfully used accelerometers to measure gait in clinical settings comparing the measurements in mobile devices to traditional medical accelerometers [43, 44, 14]. While useful, these studies are either treadmill walking studies or nurse assisted walking studies. We are instead interested in unconstrained free walking or walking without a set pace or speed. This is necessary to determine the information leaks contained in mobile devices as subjects walk around in their ordinary lives. We therefore validate the ability of the phone to act as a medical accelerometer and design software to collect medical quality sensor readings. We then conduct clinical trials to collect unconstrained free walking and demonstrate the ability of cell phones to collect measurements relating to the health of the subjects.

3.1 Medical Quality Readings from Smartphones

3.1.1 Hardware Comparison

It might be thought that the sensor chips found in smartphones would not be comparable to medical accelerometers; however, the accelerometer chips in smartphones are often similar or better than the accelerometers used in medical devices. Accelerometers measure the relative acceleration of the device's spatial motion. Three parameters are standardly used to evaluate the capability of the accelerometer: range, sensitivity, and sampling frequency.

Table 3.1: Physical Accelerometer Chips in Popular Devices

Product	*Sensor Chip*	*Range (g)*	*Sensitivity (mg)*	*Sampling*
Mobile Devices				
Apple iPad2	STMicro LIS331DLH	±2,4,8	1,2,4	.5Hz-1kHz
Apple iPad3	STMicro LIS3DH	±2,4,8,16	1,2,4,12	1Hz-5kHz
Apple iPad4	STMicro LIS3DH	±2,4,8,16	1,2,4,12	1Hz-5kHz
Apple iPhone 4S	STMicro LIS331DLH	±2,4,8	1,2,4	.5Hz-1kHz
Apple iPhone 5	STMicro LIS331DLH	±2,4,8	1,2,4	.5Hz-1kHz
Apple iPhone 6	Invensense MP67B	±2,4,8,16	.061,.122,.244,.488	400 kHz Max
Asus Eepad	Kionix KXTF9	±2,4,8	1,2,4	25Hz-800Hz
HTC Evo 4G	InvenSense MPU-9150	±2,4,8,16	.06,.1,.25,.5	4Hz-1kHz
LG Optimus Zone	Bosch BMA 222	±2,4,8,16	2,4,8,16	8Hz-1kHz
Motorola Droid 3	Kionix KXTF8	±2,4,8	1,2,4	25Hz-800Hz
Motorola Droid Mini	Kionix KXTF9	±2,4,8	1,2,4	25Hz-800Hz
Motorola Razr	STMicro LIS3DH	±2,4,8,16	1,2,4,12	1Hz-5kHz
Motorola Xoom Tablet	Kionix KXTF8	±2,4,8	1,2,4	25Hz-800Hz
Nokia 808 PureView	STMicro LIS302DL	±2,8	18,72	100Hz or 400 Hz
RIM Playbook	Bosch BMA150	±2,4,8	4,8,16	25Hz-1.5kHz
Samsung Beam	Bosch BMA220	±2,4,8,16	.02,4,63,250	32Hz-1kHz
Samsung Nexus I515	Bosch BMA220	±2,4,8,16	.02,4,63,250	32Hz-1kHz
Samsung Note	STMicro LIS3DH	±2,4,8,16	1,2,4,12	1Hz-5kHz
Samsung Galaxy Ace	STMicro LIS3DH	±2,4,8,16	1,2,4,12	1Hz-5kHz
Samsung Galaxy SIII	STMicro LIS3DH	±2,4,8,16	1,2,4,12	1Hz-5kHz
Samsung Galaxy S4	STMicro LSM330	±2,4,8,16	.061,.122,.183,.732	10MHz Max
Samsung Galaxy S5	Invensense MPU-6500	±2,4,8,16	.061,.122,.244,.488	400 kHz Max
Toshiba Thrive Tablet	Kionix KXTF8	±2,4,8	1,2,4	25Hz-800Hz
Medical Devices				
Actigraph GT3X	ADXL335	±3	2.93	.5Hz-100Hz
Dynaport Minimod	Unknown Piezo-resistive	±2,6	1,4	100Hz
Philips Actical	Unknown Piezo-electric	±2	20	32Hz
Zephyr BioHarness	Unknown MEMS	±16	16	100Hz
Fitness Devices				
Fitbit Flex	STMicro LIS2DH	±2,4,8,16	1,2,4,12	1Hz-5kHz
Jawbone Up24	Bosch BMA 250	±2,4,8,16	2,4,8,16	8Hz-1kHz

The range is the absolute minimum and maximum accelerations the chip can record in gravity units (g) with 1 g being standard earth gravity. For example, ±2 g means the accelerometer can measure a minimum of $-2 * 9.81\ m/s^2$ and a maximum of $2 * 9.81\ m/s^2$. The sensitivity is the amount of change in acceleration required to produce one bit of change on the output of the digital reading. Thus, lower numbers are better and a rating of 1 mg means that the minimum discernable difference in acceleration is 0.001 gravity units. Sensitivity is usually limited by the analog-to-digital converter embedded in the sensor chip. Finally, the sampling rate of the accelerometer is how often the sensor can be sampled in readings per second (Hz). A higher frequency rating is more preferable.

Table 3.1 lists popular mobile, fitness and medical devices along with their actual accelerometer chips (if known) and technical specifications. We identify the chips in commercial devices by combining information from tear-

downs conducted by chipworks.com with our own forensic analysis of missing devices [63]. The chip specifications are taken directly from the chip manufacturers website. Medical devices are proprietary, often less popular and are thus much less likely to be reverse engineered. Thus, the information is often incomplete; however, the available technical specifications are provided for four popular medical devices.

Table 3.1 demonstrates some interesting trends among mobile devices. The table summarizes chips used by each of the most popular phone manufacturers inside the United States including Apple, HTC, Motorola and Samsung. All the accelerometers used in phones are based on MEMS technology. The newest phones, including the Apple iPhone 6, the Samsung Galaxy S5 and the HTC Evo 4g, are utilizing a new chip manufactured by InvenSense which contains an accelerometer and a gyroscope. This chip is capable of 16 g readings as well as much higher sensitivities to smaller changes in acceleration. This new chip is being used due to breakthroughs in power requirements. The chip implements numerous low power collection techniques, interrupt capabilities to wake the device on activity change, and an on-chip proprietary step counter. Older Apple and Samsung products contain less sophisticated chips produced by STMicroelectronics capable of reading dynamically at 2 g, 4 g, 8 g, or 16 g with sensitivities of 1 mg, 2 mg, 4 mg, and 8 mg respectively. Older devices including the Beam, and RIM Playbook contain chips that cannot attain the 16g measurement instead being limited to 8 g. The accelerometers found in older phones such as the Motorola Droid, Toshiba and Asus products are manufactured by Kionix and sacrifice the higher end 16 g readings and are limited to lower sampling rates often under the 1 kHz that most other accelerometers can sample. Not included in the chart are samples from cheaper less popular phones; however, these phones also utilize MEMS based accelerometers with similar characteristics often in similar quality to the Kionix chipsets.

Moving on to the fitness devices, we note that the Fitbit uses a STMicro LIS2DH which is almost identical to the STMicro LIS3DH found in the Samsung Galazy Ace and the Jawbone Up24 uses a Bosch BMA 250 a slightly newer, lower power version of the BMA 222 used in the LG Optimus Zone. Thus, the most popular fitness devices are using the same MEMs chips in their designs which were being used in the mobile devices designed in the same time period.

While commercial devices are fairly uniform with the type of accelerometer chips, medical devices tend to be more proprietary, more closed to disclosing the enclosed hardware, and more specialized. The bottom of Table 3.1 lists some accelerometer specifications for devices which actually publish their specifications. The Actigraph GT3X and Omron Healthcare (not shown) devices both have embraced the MEMS accelerometers and market their products as utilizing cutting edge technology. Actigraph uses a ADXL335 chip commonly used in hobbyist robotics. This chip has lower capabilities of other more expensive chips but markets as a low power alternative. Validated medical devices use older accelerometer technology such as piezo-resistive or piezoelectronic accelerometers. The accelerometer used in the Philips health product Actical is limited to ± 2 g with a sampling frequency of 32 Hz. The McRoberts Dynaport Minimod employs a very impressive piezoresistive accelerometer capable of ± 2 g and ± 6 g with 1 mg and 6 mg sensitivities. This device is also limited to frequency sampling at 100 Hz. These devices utilize older technology that they claim is better than MEMS due to higher precision. However, recent studies have shown that piezo-based accelerometers suffer from larger biasing to thermal fluctuations [64]. MEMS accelerometers are less sensitive to thermal changes but instead suffer from low precision due to small differences during manufacturing; however, these errors in precision are singular to the specific chip and can be one-time calibrated away thereby fixing the manufacturing error and creating a more precise sensor [31].

Even with numerous types of accelerometers now widely available, MEMS accelerometers are quickly becoming the accepted standard for many medical devices. They offer lower power, greater range, and better sensitivity than previous technologies. At a fundamental level, the MEMS technology yields accelerometer devices with similar physical operating characteristics. Furthermore, many mobile, fitness, and medical devices are using the same chips from third party manufacturers. Even if older validated medical devices are not yet using MEMS, the embedded accelerometer chips have inferior physical operating specifications. Thus, the chips found in mid-high range smartphones of today should be adequately suited to monitor walking with the same quality assurance of the medical and fitness devices.

3.1.2 Medical Monitoring

Even though the underlying sensor chips in mobile devices are comparable to chips in dedicated devices, the software is limited by the design of the firmware and operating system API. These limitations do not affect the sensitivity or range of the readings but often limit the sampling frequency. While Android 4.4 puts increased emphasis on battery efficient continuous monitoring, the system is not designed to implement a medical quality monitor which requires fixed frequency readings. The design of the firmware between the hardware and the software API does not expose direct access to the sensors. Instead, a system wide sensor manager monitors, updates, and notifies registered applications to changing sensor readings. When an application registers a sensor handler, the system sensor event manager spawns a dedicated system thread to continuously poll the available sensors. Android defines a number of speeds with various levels of delays in microseconds as well as a FASTEST setting which is only delayed by the phone's hardware limitations. Android leaves it up to phone developers to implement drivers to access sensors leaving wide variation in maximum polling frequency and sensor behavior among devices. Thus, the sensor chips themselves may be capable of sampling at higher frequencies, but the sampling frequency may be substantially lowered by the phone's processing capability and driver implementation.

3.1.3 Frequency Sampling Requirements

While the sensitivity and range of the accelerometer readings is unaffected by the firmware and operating system, the sampling frequency can be substantially reduced. We therefore design software to implement a medical health monitor which samples at a frequency sufficient to capture a patient's movements accurately. This requires a determination of the necessary sampling frequency to measure human motion. While most medically validated devices choose to sample at 100 Hz, medical studies tend to put the range of human motion much lower at 3-5 Hz [16, 65]. Motion capture designed to capture human motion typically is calibrated to capture from 15-30 Hz [24, 25].

In order to determine the required sampling frequency, we analyze ac-

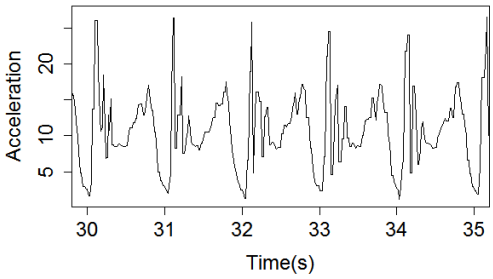

Figure 3.1: Magnitude of Walking Acceleration

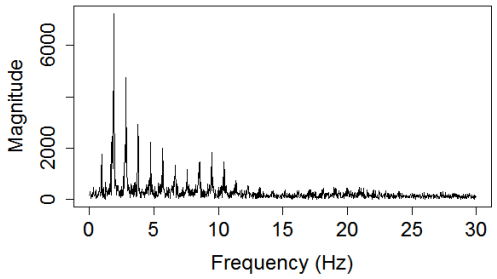

Figure 3.2: FFT of Magnitude of Walking Acceleration

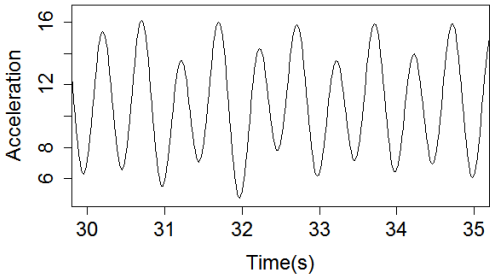

Figure 3.3: Low-Pass Magnitude of Walking Acceleration

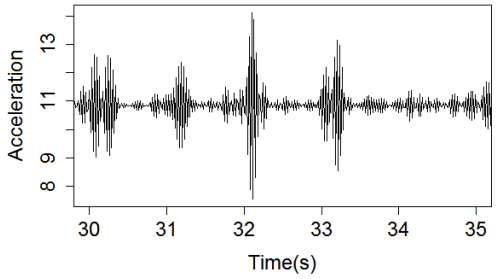

Figure 3.4: High-Pass Magnitude of Walking Acceleration

celerometer readings collected while a patient is walking. Figures 3.1 and 3.2 plot the absolute magnitude of acceleration and the corresponding Fourier transform as measured by a Samsung Galaxy S5 at 60 Hz during walking. The curve demonstrates a typical noisy signal with peaks indicating the toe-off stage of the gait pattern. The spectrograph gives little interesting signal above 15 Hz with primarily noise in the higher bands. Figures 3.3 and 3.4 illustrate the results of running the accelerometer data through a fourth-order ideal Butterworth 15 Hz high-pass and 2 Hz low-pass filter respectively. The low-pass filter yields a discernable sinusoidal signal with peaks corresponding to each step while the high-pass filter demonstrates primarily white noise. We also compare the amount of noise to good signal by considering a measured walking signal to a noise signal taken when the phone is stationary. These two measurements allow us to calculate the signal-to-noise ratio as seen in (3.1). We filter out the gravity present in the noise and walking signals. The signal-to-noise ratio of both signals after being filtered with our low-pass filter is 6.31 dB demonstrating a strong amount of actual walking signal in the lower-frequency data. The signal-to-noise ratio is calculated to be -6.96 dB for the high-pass filter. Thus, we confirm that the substantial bulk of the

walking signal is found in the lower bands and the higher-frequency bands above 15 Hz primarily contribute noise to a high-quality medical monitor. Therefore, a monitoring frequency of 60 Hz is more than sufficient to capture walking characteristics. Sixty hertz yields a Nyquist rate of 30 Hz in the output signal. This creates a medical device capable of measuring signals that occur at thirty times a second, the same frequency that the average hummingbird beats its wings while in flight. While proponents of expensive medical devices may assert this sampling frequency being too low, there is little medical evidence that such rapid movements are necessary to measure a subject's movement.

$$\begin{aligned} SNR_{dB} &= P_{Signal,dB10} - P_{Signal_Noise,dB} \\ &= log_{10}\frac{P_{Signal}}{P_{Noise}} = 20log_{10}\frac{A_{Signal}}{A_{Noise}} \end{aligned} \tag{3.1}$$

3.1.4 MoveSense

We design MoveSense, a middleware designed to overcome frequency sampling limitations on Android devices thereby transforming the phone into a medical quality monitor. It accomplishes this by implementing its own sensor reading queue. The overall design is shown in Figure 3.5. The Android sensor manager continuously queries the sensors to access the raw readings and dispatches OnSensorChanged messages to MoveSense. MoveSense spawns a high-priority thread to process the accelerometer and gyroscope sensor readings, record the timestamp and magnometer and insert this information into a first-in first-out queue. MoveSense implements a data handler thread to continuously poll the queue for new sensor readings. The data handler pops sensor readings and calculates the interval to the next reading assuming a fixed sampling frequency. If the readings are coming in at greater than the fixed frequency, the algorithm will average all readings in the current interval. If the readings are coming in at less than the fixed frequency, the algorithm will extrapolate the missing values using a linear approximation algorithm. The final output from the accelerometer pipeline is a fixed frequency sampling from both the accelerometer and gyroscope sensors. Thus, MoveSense uses the processing power in the phone to get an accurate fixed frequency

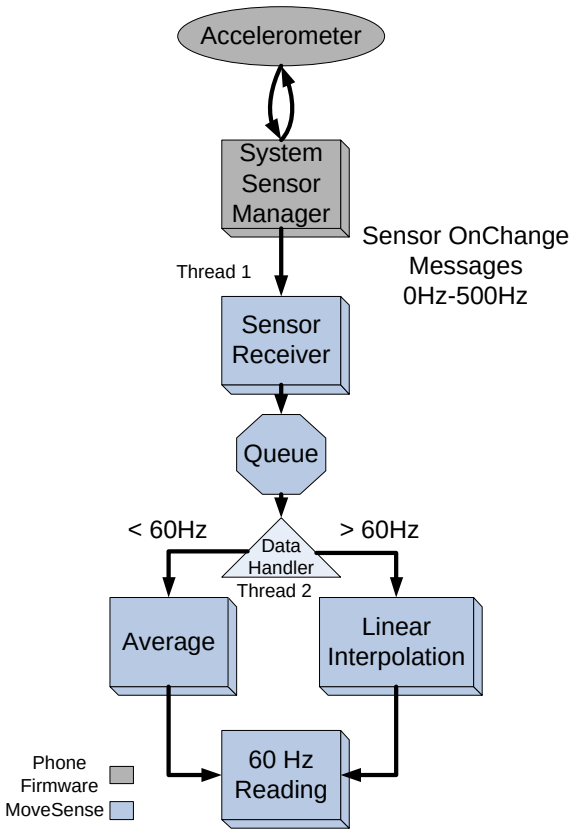

Figure 3.5: MoveSense Pipeline

stream of sensor data at the expense of processor cycles. We note that better hardware which returns sensor readings at a fixed frequency would alleviate the need for this design. We also note that a similar port of our MoveSense system could easily be ported to Apple's iOS and plan to do so in future work.

3.1.5 Validating MoveSense Readings

We test MoveSense on three phones from different vendors of various quality including a high-end Motorola Droid Mini, a mid-range last generation Samsung Galaxy Ace and a low-end LG Optimus Zone. We are interested in validating the ability of a cell phone to operate at the same accuracy as the widely accepted medical grade Actigraph GT3X.

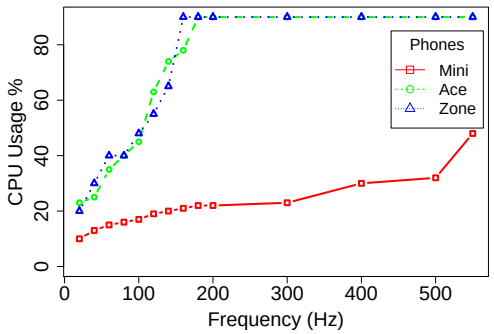

Figure 3.6: CPU Utilization versus Sensor Sampling Frequency

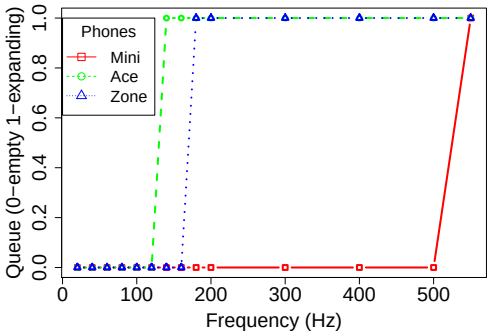

Figure 3.7: MoveSense Queue Length versus Sensor Sampling Frequency

3.1.6 Performance Benchmark of MoveSense

MoveSense must output sensor readings at a fixed frequency to allow analysis in the frequency domain. We accomplish this by sacrificing CPU cycles to attain our fixed frequency readings. The Droid Mini contains a dual core 1.7 GHz processor. Both the Galaxy Ace and Optimus Zone contain similar CPUs running at approximately 800 MHz. To determine the maximum frequency attainable with MoveSense, we measure CPU utilization while sampling the sensors at various frequencies while tethering the phones to a laptop running the Linux "top" command through the Android debugger. The top command allows us to record the CPU utilization of the MoveSense monitor. During testing, the MoveSense system records the queue length of the sensor pipeline. Figures 3.6 and 3.7 present the results of sampling at various frequency rates with MoveSense. The maximum attainable frequency is realized at the point when the CPU is fully utilized but the sensor queue does not yet grow faster than MoveSense can calculate readings. The Ace and Zone perform similarly with full utilization occurring at 120 Hz on the Ace and 160 Hz on the Zone. The Droid Mini, operating at 1700 MHz, maxes out a single CPU core (48% is a full single core utilization) at roughly 520 Hz. We notice that the ability of the phone to take readings is based on both CPU power combined with the maximum frequency the Android sensor queue reports the raw readings. When the locked frequency is higher than the incoming raw readings, more processing is required. Raw readings come into the Ace at approximately 90 Hz, the Zone at approximately 110 Hz, and the Mini at approximately 350 Hz. We finally note that the dual core

Table 3.2: Analysis of Variance (ANOVA) between Smartphones and an Actigraph GT3X

Phone	μ^2	Avg μ^2 Residual	Pr(>F)	p-value
S5	15.132	1.014	1.114e-4	.001
Ace	15.508	1.014	9.36e-5	.001

processor in the Mini would allow the user to multitask during monitoring. All phones can easily attain both our target for human motion of 60 Hz and the standard accepted 100 Hz set by medical monitors.

Finally, we test MoveSense against an Actigraph GT3X on two phones including a high-end Samsung S5 and low-end Samsung Galaxy Ace. We are interested in validating the ability of the two smartphones to operate at the same accuracy as the Actigraph GT3X, a medical grade device [66]. We measure a subject walking 10 laps on a 10 meter hallway while wearing a waist belt at the L3 position (at the base of the back) [16] containing the Actigraph and the two test phones running MoveSense. The 10 laps yield 3329 data points. We conduct an analysis of variance (ANOVA) between each of the signals from the phone and Actigraph using the signal's Z-scores. The results are summarized in Table 3.2. We find that the phones pass the F-test with probabilities of differing of $1.14e^{-4}$ and $9.36e^{-5}$ for the S5 and Ace respectively using a confidence interval of .001. Therefore, the readings from the phones are comparably identical to the readings of the Actigraph.

3.2 Clinical Trials

In order to prove the viability of inferring medical information from spatiotemporal sensor readings collected with phones being carried by patients, we conduct two clinical field trials. We choose to test patients with chronic obstructive pulmonary disease (COPD) since these patients are regularly assessed by conducting an American Thoracic Society (ATS) standard six-minute walk test (6MWT). The 6MWT is a timed test conducted on a straight corridor ranging from 10-100 meters with the ends marked with cones. Vital signs are taken at the start and end of the test including heart rate, blood oxygen levels and questionnaires including the modified Borg dyspnea and the Borg rating of perceived exertion. The subject then walks back

and forth on a straight walkway for six minutes being instructed to walk as much as they can during the test. We make no changes to the standard protocol [67] except that we make subjects carry phones running the MoveSense monitor. Subjects are given a waist pack containing the phones and instructed to wear it at the L3 position (at the base of the back) throughout the entire testing procedure.

We now outline results from two clinical trials conducted while collecting spatiotemporal data from cell phones. Our first trial collected data from thirty patients with either asthma or COPD tested at the University of Illinois Health and Hospital system in Chicago under the supervision of the University of Illinois at Chicago Institutional Review Board protocol #2011-0625. This trial collected data from patients as they walked a 6MWT while being supervised by a clinician. The second trial measured twenty-eight patients who had available spirometer data at the Carle Rehabilitation Center in Urbana Illinois under the supervision of the Carle Foundation Hospital Institutional Review Board protocol #497221. This trial also collected data during a standard 6MWT with an additional optional free walk around an oval course allowing the collection of a second sample of walking while not under testing conditions. Five healthy subjects contributed walking data to compare fitness devices to MoveSense in our laboratory in Siebel. Finally, thirty additional subjects have contributed data by running MoveSense on their phones.

In each of these trials, we investigate the ability to predict health and medical values by training machine learning algorithms using eight spatiotemporal features extracted from the raw signal chosen based on previous kinesiology research [44]. We calculate these statistical parameters using the absolute magnitude of the acceleration of the three directions. In the time domain, we select the mean (μ) and standard deviation (σ) of acceleration. We also select the mean crossing rate (MCR) which represents the ratio of above and below the mean acceleration.

$$\mu = \frac{1}{N} * \sum_{t=0}^{N} a_t \tag{3.2}$$

$$\sigma = \sqrt{\frac{1}{N} * \sum_{t=0}^{N} (a_t - \mu)^2} \tag{3.3}$$

$$MCR = \frac{1}{N} * \sum_{t=1}^{N} I_t\{I_t = 1|(a_{t-1} - \overline{a})(a_t - \overline{a}) < 0\} \tag{3.4}$$

The root mean square (RMS) provides a statistical measure on the variation of signal magnitude.

$$RMS = \sqrt{\frac{1}{N} * \sum_{t=0}^{N} a_t^2} \tag{3.5}$$

The Autocorrelation coefficient (AC) measures periodical similarity in the time domain.

$$AC = max\ \{\forall_{t \in N, i<t}\ (\sum_{t=0}^{N} a_t \overline{a}_{t-i}) \tag{3.6}$$

The coefficient of variance (CV) is a normalized measure for dispersion of the discrete samples.

$$CV = \frac{\sigma_a}{\mu_a} \tag{3.7}$$

In the frequency domain, we compute the peak frequency (PF) which represents the frequency of the peak magnitude in the spectrum.

$$PF = arg\, max_f\ \{x_f | f = 0, ..., N\} \tag{3.8}$$

Finally, we calculate the Shannon entropy (H) which quantifies the information contained in the acceleration spectrum.

$$H = -\sum_{f=0}^{N} x_f * log\, x_f \tag{3.9}$$

We call these eight parameters our feature extraction approach (FEA) and use these features in each of our trained machine learning models in the rest of this chapter.

3.2.1 Speed and Distance during Natural Walking

We investigate the possibility of using the phone sensors to predict both walking speed and walking distance. The ability to accurately measure walking distance would allow a standard 6MWT to be conducted outside a clinic. While previous work in health has primarily concentrated on constrained walking with patients either walking on a treadmill or walking with a pace setter (such as walking next to a nurse), we measure free walking during a 6MWT. We now summarize the results from the trials conducted at the University of Chicago as outlined in our previous publications [68, 69].

We first attempt to predict gait speed through supervised learning trained with a support vector machine (SVM) model using a linear kernel to map features. The model is trained using the FEA as input vectors and the average lap speed as the target using data from twelve subjects. We train our first model with six COPD patients from the hospital in Chicago and six healthy subjects recruited at the Institute of Genomic Biology at the University of Illinois in Urbana-Champaign. The model produces an error rate of 6.11% with personalized training using leave-one-lap-out training and 9.98% for cross validation with leave-one-subject-out training. The distance errors are given in Figures 3.8, 3.9 and 3.10. Not surprisingly, the personalized model performs with high accuracy with the leave-one-out and cross validation producing higher rates of error.

While initially promising, conducting an automated walk test requires distance not speed. Once patient testing was complete, we developed a strategy for walk distance estimation more closely mirroring popular fitness devices. First, we count the number of steps taken using our novel step detection algorithm [68] then we multiply by an estimation of the patient's stride length. We once again develop an SVM model to predict stride length. The model is trained on the thirty subjects tested in Chicago using the FEA as input vectors with the target being the estimated stride length found by dividing the lap length by the number of steps observed over the course of the lap. The distance model attains a 5.87% error rate without training and a 4.82% error rate with training when trained and validated on the results from the 6MWT. The Bland-Altman plot in Figure 3.11 demonstrates the error rates of each test and indicates little bias in the method.

We expand our analysis with the data from the Carle subjects by adding

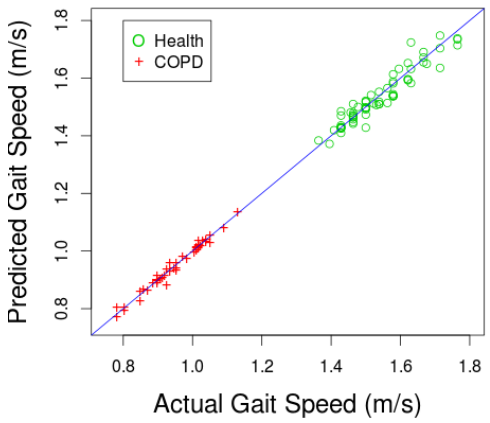

Figure 3.8: Gaitspeed Accuracy Using the Personalized Model with UIHH Patients

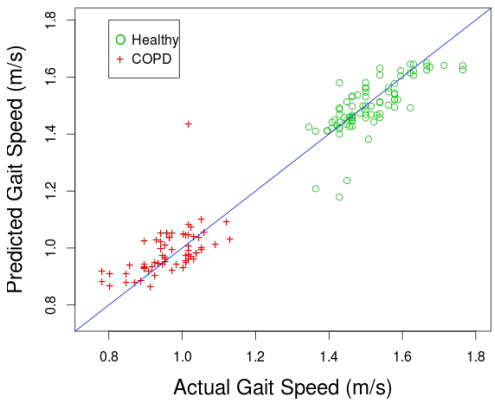

Figure 3.9: Gaitspeed Accuracy Using Leave-One-Out Validation with UIHH Patients

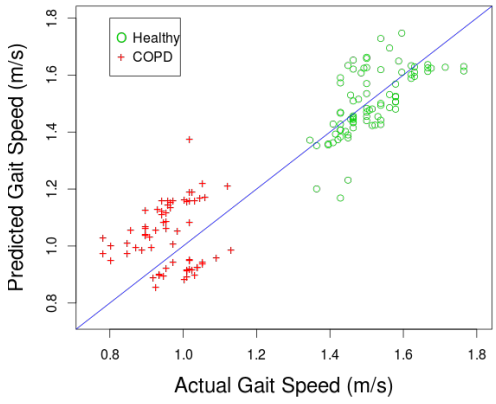

Figure 3.10: Gaitspeed Accuracy Using Cross Validation with UIHH Patients

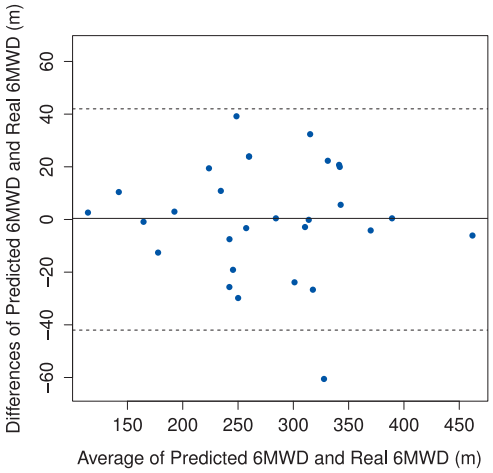

Figure 3.11: Bland-Altman Plot of Distance Estimation with UIHH Patients

a free walk around a 120 meter oval course in addition to the measurements of walking speed and distance during a 6MWT [26, 70]. The accuracy of multiplying the estimated stride length by the number of strides used in the Chicago experiment is limited since it is not possible to attain the instantaneous stride length per step without some form of motion capture or gait mat. Instead, we return to using a trained machine learning model for speed. We then assess two methods to predict distance based on the predicted speed. We exploit the constraints of the 6MWT by using the compass

Table 3.3: Error Rates for Speed Prediction (*CV Cross Validation) for Both FEA and CEA Input Features with Carle Patients

	FEA			CEA		
	GPR	SVM	ANN	GPR	SVM	ANN
Error	5.61%	3.23%	159.3%	9.20%	8.68%	75.73%
RMS	7.15%	4.55%	166.1%	12.45%	12.35%	93.41%
Error CV*	6.67%	3.68%	17.03%	11.33%	11.53%	21.71%
RMS CV*	8.93%	5.11%	21.80%	15.47%	16.06%	28.34%

to determine the beginning and ending times of the individual laps. We can then use our trained speed model to predict the speed during the lap and multiply the speed by the total lap time. We call this the laps method. In the second method, we split the walking into uniform ten second windows, use our trained speed model to predict the speed for each window and multiply by ten seconds to obtain the distance. We call this the free distance method. Since the free estimate does not require laps, it has the advantage of being usable outside of a 6MWT and can therefore be used for unconstrained walking.

In order to compare our speed models to previous work, we compare three commonly used machine learning algorithms to predict walking speed including support vector machines (SVM), Gaussian process regression (GPR) and artificial neural networks (ANN). We train each speed model using a feature extraction approach (FEA) which uses nine spatiotemporal parameters (our initial eight in the original FEA plus step counts using our step counting algorithm) and a component extraction approach (CEA) which uses the top fifty eigenvalues after conducting a principal component analysis of the magnitude of the Fourier transform, a technique commonly seen in the literature. The results given in Table 3.3 and Figure 3.12 demonstrate that the SVN trained using the FEA with an error rate of 3.23% was clearly superior to the SVN trained using the CEA with an error rate of 8.68%. The SVN and GPR produced less error than the ANN models. We therefore continue using the SVN with the FEA in the rest of the experiments in this chapter.

Figures 3.13 and 3.14 show the accuracy and error of each method during the 6MWT. Overall, the laps method improves accuracy lowering error rates to 2.33% with healthy subjects and 2.58% for COPD patients with personalized training and 3.10% for healthy subjects and 3.47% for COPD patients

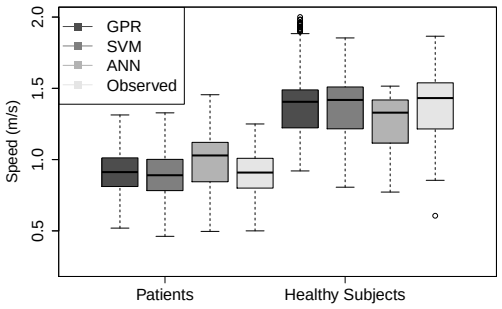

Figure 3.12: 6MWT Speed Accuracy from Carle Trial

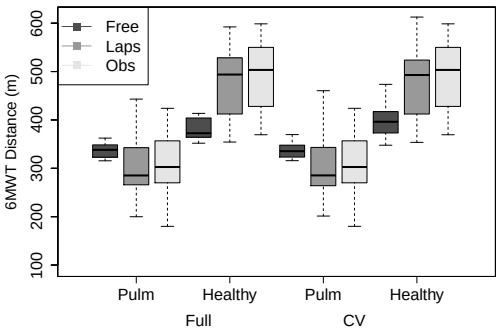

Figure 3.13: 6MWT Distance Accuracy from Carle Trial

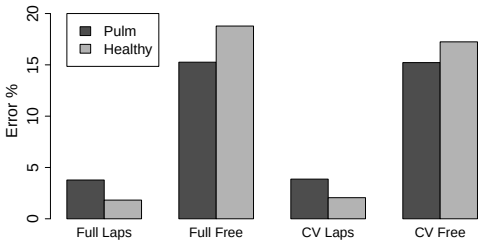

Figure 3.14: 6MWT Distance Estimation Error from Carle Trial

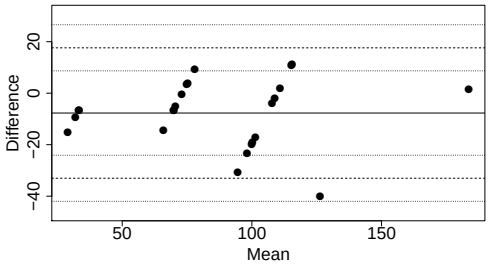

Figure 3.15: Free-Walk Distance Bland-Altman Plot from Carle Trial

without personalized training. The free method produces higher errors with a 10.97% error rate for healthy and an 11.35% error rate for pulmonary patients with personalized training and 11.28% for healthy and 11.92% for pulmonary without personalized training. Figure 3.15 presents the Bland-Altman plot for the distance prediction during the free walk around the oval. The method is biased to predict low by roughly 7.2 meters. The method results in a 10.2% error rate with a 7.6% standard deviation. Analysis indicates that a majority of the error is occurring due to the decrease in speed as the patients are coming into the turn during the 6MWT. Since the patients do not slow down while walking around the oval, the method predicts lower than the actual distance. This motivates the need for instantaneous gait speed to train the models. We leave this as an important topic to future work, but do note that medically valid gait speed and distance does indeed seem possible using machine learning methods.

Table 3.4: SVM Gold Classifier

	Unified Prediction (22 Patients)		
Actual Status	GOLD1	GOLD2	More Sever
GOLD1	**78.54%**	19.51%	1.95%
GOLD2	19.96%	**65.18%**	17.85%
More Severe	2.96%	23.47%	**73.57%**
	Cohort Prediction (12 Patients)		
Actual Status	GOLD1	GOLD2	More Sever
GOLD1	**99.24%**	0%	0.76%
GOLD2	0%	**87.63%**	12.37%
More Severe	1.23%	11.6%	**87.16%**

3.2.2 COPD Status

During our analysis of walk speed and distance with the patients tested in Chicago, we noted that walk distance seemed indicative of COPD status as seen in Figures 3.8, 3.9 and 3.10. We therefore train an SVM binary classifier using the FEA as inputs with the target being COPD or non-COPD status. This model attains 100% classification accuracy with an initial sample of twelve patients providing an early indication that it is possible to extract COPD status from motion sensors.

After testing was completed on the thirty patients at the University of Illinois Health and Hospital system in Chicago, we revisited health status prediction [69]. We develop an SVM trained model to predict GOLD status classification for COPD patients. GOLD status classifies patients based on FEV1% values. FEV1% is a measurement of the forced expiratory volume over the first second of exhalation divided by the standard expected for the subject's demographic cohort. The GOLD classification is based solely on FEV1% values with GOLD 1 being greater than 80%, GOLD 2 being 50-79%, GOLD 3 being 30-49% and Gold 4 being $<= 29\%$. The results in Table 3.4 yield an overall 71.57% classification accuracy for a single model across all demographic cohorts and 89.22% accuracy when trained on a specific cohort. Interestingly, these results do not simply mirror the walk distance

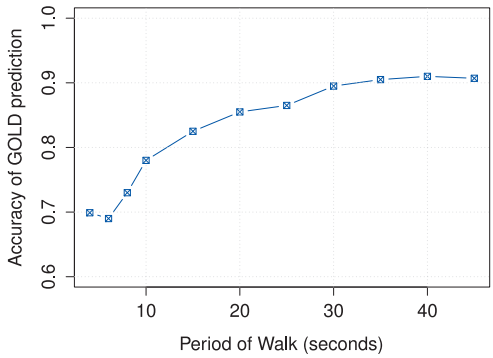

Figure 3.16: GOLD Prediction Accuracy over Time

in the groups since the average walk distance in the more severe group is 232.8 meters which is greater than the average walk distance in the Gold 2 group of 208.7 meters. Thus, severity prediction by gait parameters produced better health classification than traditional 6MWT distances. Figure 3.16 demonstrates the amount of monitoring time required to make the GOLD prediction with close to 85% accuracy after 20 seconds and 90% accuracy after 30 seconds. These tests demonstrate that we can not only train classifiers to identify the presence of COPD, but we can also train a classifier capable of determining a three way severity classification of COPD patients. Clearly, the viability of such a prediction elevates the need to protect motion data while a COPD patient is walking in order to protect leaking the status of the patient's condition.

3.2.3 Blood Oxygen Saturation

Finally, we investigate the ability to predict blood oxygen saturation continuously as the users conduct a 6MWT and free walk during our experiments at the Carle Pulmonary Rehabilitation Center [71, 72]. We find that we can predict with an error rate of roughly 1% during the 6MWT and 2% during free walking using an SVM trained using FEA on fifteen patients. The SVM using FEA was once again proven superior to Gaussian process regression and the CEA. We also find no significant difference between using the magnitude of acceleration versus the three directional components. Using cohorts, we can reduce the error by roughly one half. Thus, preliminary results indicate that walking signal contains information correlated to blood oxygen satura-

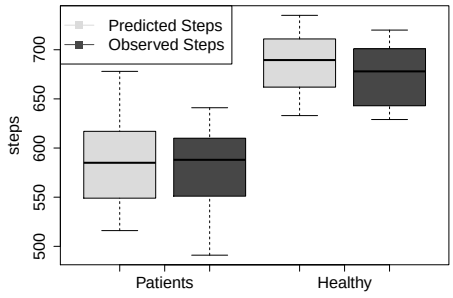

Figure 3.17: MoveSense Step Counting Accuracy

Table 3.5: Step Counting Accuracy of MoveSense vs. Commercial Devices

Device	Error Rate
Phone App	0.94%
Omron HJ-720ITC	5.20%
Actigraph GT3X	11.08%

tion, a widely used medically valid metric for patient status and clearly a medically sensitive metric.

3.2.4 Comparison to Dedicated Devices

We investigate the ability of smartphones against a number of dedicated medical accelerometers to measure both step counts and distance walked. For the step counts, we use our novel step counting algorithm as introduced in our previous work [68]. Figure 3.17 is the boxplot for the measured number of steps compared to the actual number of steps taken for fifteen patients tested at the Carle Rehabilitation center and ten healthy subjects all conducting a 6MWT. Overall, we attain 98.7 % accuracy with the COPD patients and 97.3 % accuracy with the healthy controls. Additionally, we conduct a test with four subjects wearing a phone running MoveSense, an Omron pedometer and an Actigraph GT3X. Each subject conducts a 500 step walking test. Table 3.5 presents the results with the phone app attaining 99.04% accuracy beating both the Omron and the Actigraph. Thus, phones are capable of recording accurate step counts with greater accuracy than dedicated devices on healthy subjects.

We now compare the distance estimation from MoveSense against common fitness and health devices. To directly compare, we conduct testing on five healthy subjects with one female and four males from the ages of 21 to 60 years old. Due to the cumbersome nature of carrying six devices, the IRB would not allow us to test on pulmonary patients. The subjects carry

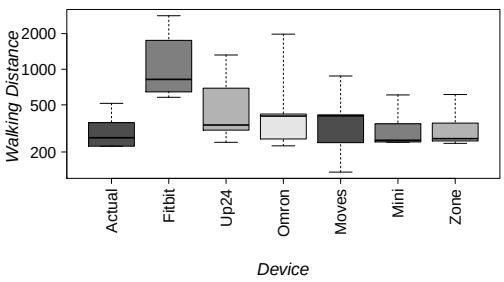

Figure 3.18: Distance Prediction Accuracy of MoveSense against Commercial Devices

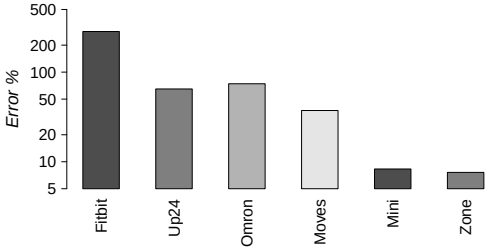

Figure 3.19: Distance Prediction Error MoveSense against Commercial Devices

a Fitbit Flex and Jawbone Up24 on their wrist, an Omron HJ-720ITC pedometer on their belt and an iPhone 5s running Moves (a commercial app from ProtoGeo now a FaceBook subsidiary), a Motorola Droid Mini running MoveSense and an LG Optimus Zone running MoveSense in a pack at the L3 position. Subjects walked between two to five laps consisting of 65 meters of busy hallways to simulate real world conditions. Subjects then stopped and conducted office tasks for three minutes at a desk and spent two minutes simulating eating a meal at a table. Finally, the subjects walked through the 65 meters of hallways two more times and ascended and descended a set of stairs. Each fitness device was calibrated with the subjects personal stride length and demographic information. We recorded the distance reported from each device and the distance from our previously trained universal free distance model for both phones running MoveSense. These were then compared to the known measured distances.

The results from the experiment are shown in Figures 3.18 and 3.19. Using the FEA SVM model trained on a mixture of COPD patients and healthy controls, we attain an error rate of 8.3% from the readings on the Droid Mini and 7.6% from the readings on the LG Optimus Zone. This is substantially better than the roughly 50% error using Moves, the Up24, and Omron pedometers and much better than the 400% error produced by the Fitbit Flex. Thus, our models improve the state of the art at predicting walk speed and distance using only motion sensors during unconstrained walking with higher accuracy than the tested dedicated fitness devices.

3.3 Conclusions

Through careful analysis of the hardware, design of software, and direct testing, we find that sensors contained within smartphones can attain raw readings statistically identical to popular medical grade devices. Machine learning techniques presented in previous work are directly applicable to patients when re-trained using data from the correct target population. Walk speed and distance can be inferred from the motion sensors in smartphones with patients during unconstrained walking. Phones using a trained support vector machine model using eight input statistical features chosen from the kinesiology literature is the most accurate model tested in our experiments for walk speed and walk distance. Our distance model lowers error rates compared to consumer grade fitness devices by nearly an order of magnitude. Additionally, we can infer health status including a binary classification between COPD and healthy subjects, a three way classification of severity in COPD patients, and a continuous prediction of blood oxygen saturation accurate with a mere 2% error. Thus, health status in patients with chronic obstructive pulmonary disease is classifiable using only the motion sensors in mobile devices. Of course, all these inferences need validation with larger training and testing populations. This future work is currently in progress; however, the initial proof that the models work clearly motivates the need to investigate privacy protection to mitigate the threat of leaking sensitive medical information.

CHAPTER 4

A DATA SET TO STUDY MOTION PRIVACY

Previous work indicates it is possible to infer various activity, fitness, health and demographic properties by using machine learning models with data from motion sensors. We now leverage all the phone data we have collected over a four year period of clinical trials to investigate the privacy implications of sharing motion data. Our data set includes roughly forty-two hours of readings from ten different phones testing eighty-eight subjects. It contains thirty subjects tested at the University of Illinois Health and Hospital System in Chicago and twenty-eight subjects tested at the Pulmonary Rehabilitation Center in Urbana. The remaining thirty subjects are healthy controls recruited from our lab at the Institute of Genomic Biology and support staff at Carle Hospital. Many data sessions were conducted with three phones collecting simultaneously allowing us to compare the phone classification independently of the subject and particular test. We now outline what prediction targets we will investigate and how we will extract features for the machine learning in the subsequent chapters.

4.1 Prediction Targets

Determining the vulnerabilities created by a user sharing motion data requires a careful analysis of what can be inferred from the data. We compile a list of all known information we have about the subjects in our master data set. This includes some new demographic predictions such as height, weight and age as well as revisiting previously confirmed classifiers such as phone and personal identification. While we expect to be able to confirm previous work with these classifiers, it is necessary to conduct our own analysis in order to determine the relative privacy leaks for each sensor and feature used in the analysis.

Table 4.1: Available Data per Prediction Target

	Target	Subjects	Phones	Sessions	Segments	Time
Biometrics	PhoneID All	88	10	242	1120	42:49:15
	PhoneID Walking	88	10	242	455	23:50:36
	PhoneID Idle	86	10	240	665	18:58:39
	UserID All	88	10	242	1120	42:49:15
	UserID Walking	88	10	242	455	23:50:36
	UserID Idle	86	10	240	665	18:58:39
Fitness	Speed Walking	79	9	178	265	18:27:24
	Speed Waking (Laps)	79	9	178	5956	18:18:10
	Steps Walking	8	6	24	24	2:23:00
	Steps Walking (Laps)	8	6	24	804	2:23:00
	Activity	88	10	242	1120	42:49:15
	Activity (Laps)	88	10	242	6883	42:50:37
Demographics	Gender All	88	10	242	1120	42:49:15
	Gender Walking	88	10	242	455	23:50:36
	Gender Idle	86	10	240	665	18:58:39
	Age All	86	10	239	1089	42:24:24
	Age Walking	86	10	239	441	23:33:55
	Age Idle	84	10	237	648	18:50:29
	Height All	78	10	216	992	36:52:44
	Height Walking	78	10	216	404	22:25:03
	Height Idle	76	10	214	588	14:27:41
	Weight All	78	10	216	992	36:52:44
	Weight Walking	78	10	216	404	22:25:03
	Weight Idle	76	10	214	588	14:27:41
Medical Measures	FEV1 All	64	9	123	590	20:33:25
	FEV1 Walking	64	9	123	250	13:23:57
	FEV1 Idle	62	9	121	340	7:09:28
	FEV1/FVC All	26	7	80	401	13:02:45
	FEV1/FVC Walking	26	7	80	167	9:35:04
	FEV1/FVC Idle	26	7	80	234	3:27:41
	GOLD All	85	10	237	1087	42:24:04
	GOLD Walking	85	10	237	441	23:33:56
	GOLD Idle	83	10	235	646	18:50:08

Our data set contains twelve labeled targets summarized in Table 4.1. We manually labeled the data in order to provide ground truth for each prediction target. We split certain labels between walking and non/walking segments allowing us to measure the accuracy of classifiers both when the subjects are stationary and moving. Additionally, speed and steps contain the notion of laps in order to use the speed and step algorithms presented in Chapter 3. Due to the varied nature of the data set, each target characteristic may have

various amounts of available data. The number of subjects, testing sessions, testing segments and total recorded time for each target characteristic is also recorded in Table 4.1. We consider both phone identification and user identification as our biometrics. We have walking, non-walking, speed and steps for our fitness metrics. We consider various demographics including gender, age, height, and weight. These targets will be used to assess the ability of classifiers to use motion data to divide subjects into demographic cohorts. Finally, we look at the percentage of vital capacity expired in the first second of forced exhalation (FEV1/FVC), normalized FEV1/FVC to the standard value for the subject's demographics (FEV1%), and standard COPD status classification (GOLD) as targets for our medical evaluation.

4.2 Sensor Streams

We extract every sensor stream available from current generation mobile phones giving thirty-one streams of sensor input as presented in Table 4.2. Values 1-19 are obtained directly from the phone's sensors by MoveSense. MoveSense queries each value from the Android API. The Accelerometer, Magnetometer, Gyroscope and Gravity X, Y, and Z values come directly from the built-in sensor chip. The gravity reading is the DC component of the acceleration from a low-pass filter. The rotation vector is calculated by the Android API using the gravity and magnetometer sensors to estimate the orientation of the phone in 3d space using (4.2). The rotations are calculated using a world coordinate system with the positive X direction pointing east and the rotation about X in the clockwise direction given by θ. The Y direction points north with rotation around Z in the clockwise direction given by ψ. The Z direction points up perpendicular to the ground with positive rotations in the clockwise direction given by ϕ. Note that this is different from the traditional right-hand coordinate system used in aviation. These sensors are standard sensors as implemented by the Android API and are collected at maximum speed and sampled at a fixed 60 Hz frequency by our MoveSense system as outlined in Chapter 3. We also add to this list the continuous streams of heart rate and blood oxygen level that was collected through a wireless pulse oximeter during the clinical testing.

Table 4.2: Sensor Streams

1	AccX	Acceleration in X Direction (Phone Coordinates)
2	AccY	Acceleration in Y Direction (Phone Coordinates)
3	AccZ	Acceleration in Z Direction (Phone Coordinates)
4	AccMag	Magnitude of Acceleration (4.1)
5	MagX	Magnetic Field Strength in X Direction (Phone Coordinates)
6	MagY	Magnetic Field Strength in Y Direction (Phone Coordinates)
7	MagZ	Magnetic Field Strength in Z Direction (Phone Coordinates)
8	MagMag	Magnitude of Magnetometer (4.1)
9	GyroX	Rotational Velocity around X (Phone Coordinates)
10	GyroY	Rotational Velocity around Y (Phone Coordinates)
11	GyroZ	Rotational Velocity around Z (Phone Coordinates)
12	MagGyro	Magnitude of Rotational Velocity (4.1)
13	GravX	Magnitude of Gravity in X Direction (Phone Coordinates)
14	GravY	Magnitude of Gravity in Y Direction (Phone Coordinates)
15	GravZ	Magnitude of Gravity in Z Direction (Phone Coordinates)
16	MagGrav	Magnitude of the Gravity Vector (4.1)
17	RotX	Rotation Azimuth (World Coordinates ϕ (4.2))
18	RotY	Rotation Pitch (World Coordinates θ (4.2))
19	RotZ	Rotation Roll (World Coordinates ψ (4.2))
20	HR	Heart Rate Measured via BlueTooth PulseOx
21	POx	Blood Oxygen Saturation Measured via BlueTooth PulseOx
22	AccV	Acceleration in Vertical Z Direction (World Coordinates)
23	AccE	Acceleration in the East X Direction (World Coordinates)
24	AccN	Acceleration in the North Y Direction (World Coordinates)
25	AccF	Acceleration in the Forward Walking Direction
26	AccSw	Acceleration Perpendicular to Forward Walking Direction
27	AccPCA1	Acceleration in Direction of Maximum Variance
28	AccPCA2	Acceleration Perpendicular to Maximum Variance
29	DirF	Rotation from North of Walking Direction in Degrees
30	DirP	Rotation from North of Maximum Variance in Degrees
31	DirOff	Absolute Difference between DirF and DirP

4.2.1 Calculated Sensor Streams

Sensor streams 22-31 are calculated from the other sensor readings collected from MoveSense. For each of the raw sensor streams, we calculate the Euclidian distance for the accelerometer, magnetometer gyroscope, and gravity readings (4.1). We also conduct three projections of the accelerometer readings. We project the accelerometer values to a standard world coordinate system. We also project the accelerometer values with the X axis in the

forward walking direction. Finally, we apply a transformation with the X axis aligned with the vector of maximum variance during walking. We also calculate the forward walking direction and maximum variation in degrees from magnetic north.

We project the accelerometer readings from the local phone coordinate system to the world coordinate system. This transformation uses the values of the magnetometer M which give the vector north and the gravity reading G which gives the vector up opposite of gravity (4.2). The gravity vector is assumed to be more accurate at pointing up than the magnetometer is at pointing north. Thus, the cross product of the gravity vector and magnetometer are first taken to estimate the heading east H. The eastward vector and gravity are normalized. The direction north is then taken as the cross product of the gravity and the eastward vector to get the north vectors. The local to world coordinate systems can then be transformed by simply multiplying the given direction by the rotational matrix. The order of rotation for the yaw, pitch and roll are to first rotate about the Z (gravity) axis, then about the X (eastward) axis and finally about the Y (northward) axis. The final definition of the rotation vector in terms of the roll pitch and yaw and the equations to obtain them directly from the rotation vector are given in (4.2). The values of AccV, AccE, and AccN are then the output from multiplying the AccX, AccY and AccZ by the rotation matrix giving the acceleration in the vertical, east and north directions respectively.

$$|V| = \sqrt{{V_X}^2 + {V_Y}^2 + {V_Z}^2} \tag{4.1}$$

$$Rot = \begin{bmatrix} R_0 & R_1 & R_2 \\ R_3 & R_4 & R_5 \\ R_6 & R_7 & R_8 \end{bmatrix} = \begin{bmatrix} \frac{H_X}{|H|} & \frac{H_Y}{|H|} & \frac{H_Z}{|H|} \\ \frac{M_X}{|M|} & \frac{M_Y}{|M|} & \frac{M_Z}{|M|} \\ \frac{G_X}{|G|} & \frac{G_Y}{|G|} & \frac{G_Z}{|G|} \end{bmatrix} =$$

$$\begin{bmatrix} cos\phi & sin\phi & 0 \\ -sin\phi & cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & sin\theta \\ 0 & -sin\theta & cos\theta \end{bmatrix} = \begin{bmatrix} cos\psi & 0 & sin\psi \\ 0 & 1 & 0 \\ -sin\psi & 0 & cos\psi \end{bmatrix}$$

$$\begin{bmatrix} cos\phi cos\psi - sin\phi sin\psi sin\theta & sin\phi cos\theta & cos\phi sin\psi + sin\phi cos\psi sin\theta] \\ -(sin\phi cos\psi + cos\phi sin\psi sin\theta) & cos\phi cos\theta & -sin\phi sin\psi + cos\phi cos\psi sin\theta) \\ -sin\psi cos\theta & -sin\theta & cos\psi cos\theta \end{bmatrix}$$

$$\phi = tan^{-1}\left(\frac{R_1}{R_4}\right)$$

$$\theta = sin^{-1}(-R_7)$$

$$\psi = tan^{-1}\left(-\frac{R_6}{R_8}\right)$$

(4.2)

A majority of the data in our data set is collected while the subject is walking. Measuring the amount of acceleration relative to the walking direction of the subject may allow the phone to more closely track and classify the characteristics of the gait pattern. The method we base our algorithm to estimate walking direction was first proposed by Roy et al. [27]. The forward walking direction is observed to be the average of the horizontal direction during the swing phase over an entire gait cycle. The swing phase is determined using the heel strike, which is easily identified as a spike in the acceleration magnitude, as an anchor point. Once identified, the direction measured over successive heal strikes can be averaged giving a forward walking direction.

To identify the walking direction, we filter the vertical component of acceleration through an ideal second-order Butterworth low-pass filter with cutoff frequency set at 2 Hz. This yields a roughly sinusoidal signal with the neg-

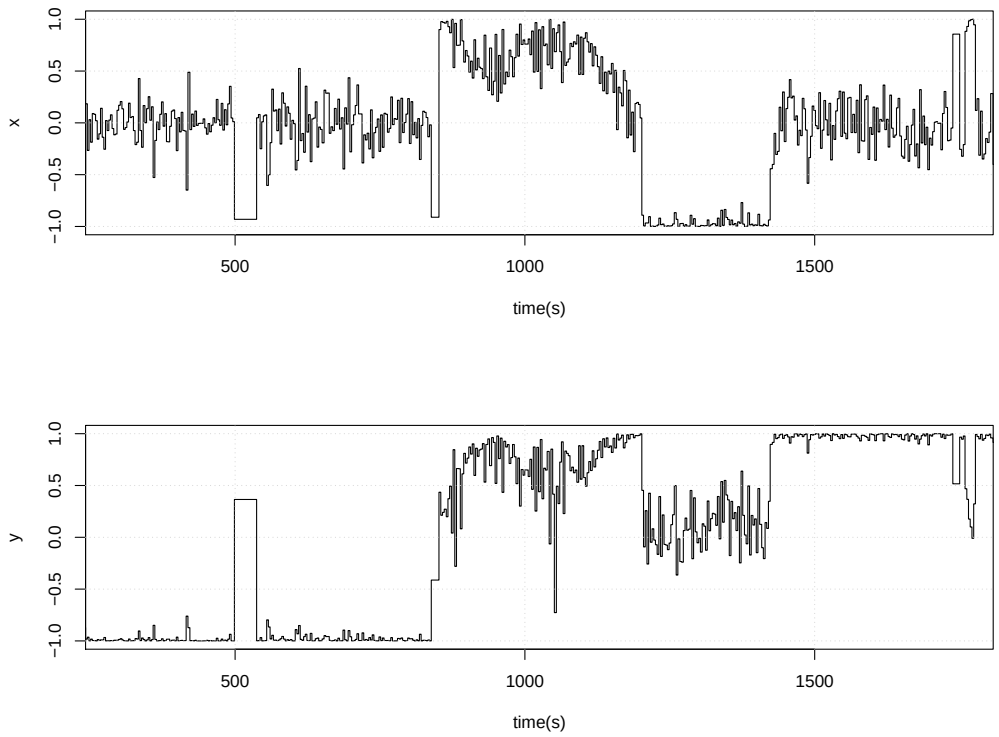

Figure 4.1: Walk Direction Test

ative magnitude roughly corresponding to the swing phase of the gait. We therefore calculate the average direction of the acceleration in the horizontal plane (AccN and AccE) while the vertical acceleration is negative to get the average forward direction per step. We average this direction over four cycles to obtain the direction per step. Figure 4.1 plots the results from a test conducted with the walk direction detection. The graph plots the relative absolute magnitude of the directional components in the AccE (X) and AccN (Y) directions during walking. In this experiment, the subject walked a two mile course with segments pointing due south, north-east, west, and finally north. We see that while the signal is noisy, we can clearly identify the direction with the south portion of walking from 0 to 850 seconds, the north-east from 850 to 1200 seconds, the west from 1200 to 1450 seconds and the north walking from 1450 seconds to the end of the experiment. The noise in the signal is trivially removed via another low-pass Butterworth filter. Finally, we implement a rotational matrix R_W to represent the transform from horizontal

north and east acceleration to forward walking direction (4.3). Multiplying the north and east directional components gives the relative acceleration in the forward walking direction accF and the acceleration in the perpendicular direction parallel to the sway during the subject's walking accS.

$$Rot_W = \begin{bmatrix} \frac{F_X}{|F|} & \frac{F_Y}{|F|} \\ \frac{F_Y}{|F|} & -\frac{F_X}{|F|} \end{bmatrix} \tag{4.3}$$

We transform the acceleration in the horizontal plain to the direction of maximum variance and the orthogonal component by conducting a principal component analysis (PCA). The direction of maximum variance is interesting because it is related to the stability of the subject throughout the gait cycle. In particular, a healthy stable individual has a tendency to have maximum variation parallel to the forward walking direction. Conversely, an unstable subject has more variance off the primary walking direction. Thus, investigating the relative direction of maximum variance during walking may provide insight into the gait cycle. We therefore assemble a rotation matrix R_{PCA} from the output of the PCA. Multiplying the acceleration in the horizontal direction yields accP1 and accP2.

Finally, we add the direction of the horizontal transforms in the horizontal plane. We calculate the directions using standard coordinates with 0° being north and the rotation going clockwise i.e. 90° is east. We calculate the direction of the forward walking direction dirF and the direction of the PCA dirP using $\theta = arctan(\frac{R_{0,1}}{R_{0,0}})$ (taking the quadrant into consideration to get an angle between 0° and 360°). Finally, we calculate the difference between the forward walking direction and the PCA direction called dirOff.

4.3 Feature Extraction

We now have thirty-one sensor streams which can be used to predict the target characteristics of interest. To use machine learning models, we must extract useful features from each of these streams. While there have been a variety of proposed features in previous work, we attempt to extract the most comprehensive list of features in order to investigate potential privacy leaks. We note that some of these features may seem redundant and/or ill-suited for this data. However, these features should demonstrate low utility

during the feature selection analysis conducted in Chapter 5. We process the data by first assembling sessions of readings with a specific target. We then iterate over every session, group the data into 512 reading sliding windows (8.5 s) with a 256 reading overlap (4.25 s). For each window, we extract 74 features in the time and frequency domain using a combination of the LibXtract toolbox [73] and custom code written in python to fill in gaps missing from the library and sometimes correcting for minute errors in the libraries. The final output is a list of windows with the target and 31 sensors multiplied by 74 features giving 2,294 sensor features for further analysis.

The first fifteen features are calculated directly from the time domain over a window X containing N samples from $(x_0, x_1...x_N)$ as shown in Table 4.3. This includes standard statistics of the window estimating the average and spread of values including the mean (4.4), the variance (4.5), the standard deviation (4.6), and the average deviation (4.7). The skewness is a measure of the asymmetry of the window's data with positive values indicating a majority of the signal is to the left of the window and a negative skew indicating a majority of the signal is to the right of the window (4.8). The kurtosis is a measure of the peakedness or sharpness of the peak of the window (4.9). A kurtosis of zero indicates the distribution is close to normal while a negative kurtosis is flatter than normal and a positive value demonstrates a distribution with a peak greater than a normal distribution. We also extract the signal minimum (4.10), signal maximum (4.11), and sum of all values over the window (4.12). The number of non-zero entries are also calculated (4.13). The root-mean-square gives another approximation of the average strength of the signal (4.14). The fundamental frequency (f_0) is estimated using the Average Magnitude Difference Function (AMDF) (4.48). If the signal contains periodicity, the shift at the fundamental frequency should be close to zero difference plus some error term caused by noise. In this method, we first extract an estimate of the baseline noise as $AMDF_1$. We then find the minimum shift τ which produces the $AMDF_\tau < AMDF_1$. This value of τ is approximately the period. We estimate the contribution of the error by dividing the $AMDF_\tau$ by $AMDF_1$. The fundamental frequency is therefore the sampling frequency F_S divided by the period (4.15). We also calculate the signal's mean crossing rate which is the average number of times the signal crosses the mean across the window (4.16). We calculate the standard Shannon entropy giving an estimate of the uncertainty in the signal (4.17).

Table 4.3: Standard Features Extracted from X Containing (x_1,x_2...x_N)

1	Signal Mean ($\bar{x}$)	$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n$ (4.4)
2	Signal Variance	$Var(X) = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \bar{x})^2$ (4.5)
3	Signal Standard Deviation (σ_x)	$\sigma_x = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(x_n - \bar{x})^2}$ (4.6)
4	Signal Average Deviation	$AvDev(X) = \frac{1}{N}\sum_{n=1}^{N}\lvert x_n - \bar{x}\rvert$ (4.7)
5	Signal Skewness	$Skew(X) = \frac{\frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})^3}{(\frac{1}{N-1}\sum_{n=1}^{N}(x_n - \bar{x})^2)^{3/2}}$ (4.8)
6	Signal Kurtosis	$K(X) = \left(\frac{\frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})^4}{(\frac{1}{N-1}\sum_{n=1}^{N}(x_n - \bar{x})^2)^2}\right) - 3$ (4.9)
7	Signal Minimum	$min(X) := \text{argmin}_{\forall n \in N}(x_n)$ (4.10)
8	Signal Maximum	$max(X) := \text{argmax}_{\forall n \in N}(x_n)$ (4.11)
9	Signal Sum	$Sum(X) = \sum_{n=1}^{N} x_n$ (4.12)
10	Signal NonZero Count	$nzc(X) = \sum_{n=1}^{N}\mathbb{I}\{x_n \neq 0\}$ (4.13)
11	Signal RMS Amplitude	$RMS(X) = \sqrt{\frac{1}{N}\sum_{n=1}^{N} x_n^2}$ (4.14)

Table 4.3: Standard Features Extracted from X Containing (x_1,x_2...x_N) (Cont.)

12	Fundamental Frequency	$\tau = min(\tau \mid AMDF_\tau < AMDF_1)$ $P = \tau + \frac{AMDF_\tau}{AMDF_1}$ $f_0 = \frac{F_S}{P}$ (4.15)
13	Signal Mean Crossing Rate	$zcr(X) = \frac{1}{N}\sum_{n=1}^{N-1} \mathbb{I}\{(x_n - \mu) * (x_{n+1} - \mu) < 0\}$ (4.16)
14	Shannon Entropy	$H(X) = -\sum_{n=1}^{N} P(x_n) log_2 P(x_n)$ (4.17)
15	Coefficient of Variation	$c_v(X) = \frac{\sigma_x}{\bar{x}}$ (4.18)

Finally, we calculate the coefficient of variation (4.18).

$$S_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \forall k = 0, ..., N-1 \tag{4.29}$$

Features 16-42 are extracted from the Fourier transform of the time domain signal conducted over the window. We apply a fast Fourier transform to give the Spectrum S_k (4.29). We then define $A = (a_1, a_1, ...a_N)$ to be the real component of the transform or the amplitude of the spectrum and $F = (f_1, f_2, ..., f_N)$ to be the imaginary or frequency of the spectrum. We extract the similar corresponding basic features listed in Table 4.4 in the frequency domain as the time domain but weight the amplitude by the relative frequency. These features include the spectral mean (4.19), the spectral variance (4.20) and the spectral standard deviation (4.21). The spectral skewness demonstrates the relative proportion of the distribution in the spectrum with

Table 4.4: Standard Spectrum Features

16	Spectral Mean ($\bar{s}$)	$\bar{s} = \frac{1}{\sum_{n=1}^{N} a_n} \sum_{n=1}^{N} f_n a_n$ (4.19)
17	Spectral Variance	$Var_s = \frac{1}{\sum_{n=1}^{N} a_n} \sum_{n=1}^{N} (f_n - \bar{s})^2 a_n$ (4.20)
18	Spectral Standard Deviation (σ_s)	$\sigma_s = \sqrt{\frac{1}{\sum_{n=1}^{N} a_n} \sum_{n=1}^{N} (f_n - \bar{s})^2 a_n}$ (4.21)
19	Spectral Skewness	$SSkew = \frac{\sum_{n=1}^{N} (f_n - \bar{s})^3 a_n}{\left(\frac{1}{\sum_{n=1}^{N} a_n} \sum_{n=1}^{N} (f_n - \bar{s})^2 a_n\right)^{3/2}}$ (4.22)
20	Spectral Kurtosis	$SK = \left(\frac{\sum_{n=1}^{N} (f_n - \bar{s})^4 a_n}{(\frac{1}{\sum_{n=1}^{N} a_n} \sum_{n=1}^{N} (f_n - \bar{s})^2 a_n)^2} \right) - 3$ (4.23)
21	Spectral Minimum Amplitude	$min_a = \arg\min_{\forall n \in N}(a_n)$ (4.24)
22	Spectral Maximum Amplitude	$max_a = \arg\max_{\forall n \in N}(a_n)$ (4.25)
23	Spectral Amplitude Sum	$Sum_a = \sum_{n=1}^{N} a_n$ (4.26)
24	Spectrum Amplitude NonZero Count	$nzc_a = \sum_{n=1}^{N} \mathbb{I}\{a_n \neq 0\}$ (4.27)
25	Spectrum RMS Amplitude	$RMS_a = \sqrt{\frac{1}{N} \sum_{n=1}^{N} a_n^2}$ (4.28)

a negative value indicating a majority of the distribution to the right of the middle, a positive value indicating a majority of the spectrum is to the left of the middle and a value of zero showing an approximately normal distribution (4.22). The spectral Kurtosis once again measures the relative strength of the peak of the spectrum compared to a normal distribution (4.23). We also extract the minimum and maximum amplitudes of the signal (4.24), (4.25) and the sum of the amplitudes in the window (4.26). The RMS of the signal measure the approximate strength of the amplitude (4.28).

Table 4.5 lists the additional features we extract from the frequency spectrum. We extract two estimates of the irregularity in the spectra (4.30) and (4.31). The irregularity estimates the amount of noise between peeks in the spectra of the amount of jitter (deviation from periodicity). The Centroid is equal to the spectral mean (4.32). The smoothness is another measure of the amount of noise between peaks with the individual amplitudes being log scaled (4.33). The spectral spread measures the amount of variance around the spectrum's centroid (4.34). The roll-off frequency is the frequency at which a certain percentage of the spectral power is contained. For our analysis, we choose the roll-off frequency to contain 95% of the signal's power. Thus, 95% of the signal's power is contained below the roll-off frequency leaving 5% of the power above the roll-off frequency (4.35). The flatness measures the distribution of the energy over the spectrum. Low values of flatness indicate a majority of the energy is concentrated in a low number of frequency bands while a higher value indicates the energy is uniformly distributed across the bands (4.36). The power is the total power in the window (4.37). The sharpness is a perceptual measure used in sound which measures the relative amount of high-to-low-frequency components in the signal (4.38). A higher sharpness indicates a high number of frequency components with relatively few low-frequency components. The spectral slope also measures the relative strength of low-frequency to high-frequency components with a negative slope demonstrating more low-frequency components and a positive slope demonstrating stronger high-frequency components to the signal (4.39). The Tristimulus measures the ratios of power between bands of harmonics. Each harmonic is a multiple of the fundamental frequency estimated in the signal. Tristimulus 1 measures the ration between the first harmonic to the total (4.40). Tristimulus 2 measures the ratios of harmonics two, three and four to the total and Tristimulus 3 measures the ratio harmonics 5 and

Table 4.5: Other Spectrum Features

26	Irregularity Jensen	$I_J = \frac{\sum_{n=1}^{N}(a_n - a_{n+1})}{\sum_{n=1}^{N} a_n^2}$ (4.30)
27	Irregularity Krimphoff	$I_K = \sum_{n=2}^{N-1} \lvert a_n - \frac{a_{n-1} + a_k + a_{n+1}}{3} \rvert$ (4.31)
28	Centroid	$C = \frac{\sum_{n=1}^{N} f_n a_n}{\sum_{n=1}^{N} a_n}$ (4.32)
29	Smoothness	$Smooth = \sum_{n=1}^{N-1} \lvert 20log(a_n) - \frac{20log(a_{n-1}) + 20log(a_n) + 20log(a_{n+1})}{3} \rvert$ (4.33)
30	Spread	$Spread = \sqrt{\sum_{n=1}^{N} \left((f_n - \bar{s})^2 \frac{a_n}{\sum_{n=1}^{N} a_n} \right)}$ (4.34)
31	Rolloff Frequency	$f_r = argmin(r) \forall r \in \left\{ N \lvert \sum_{n=1}^{r} a_n^2 \geq 0.95 \sum_{n=1}^{N} a_n^2 \right\}$ (4.35)
32	Flatness	$F = \frac{\sqrt[N]{\prod_{n=1}^{N} a_n}}{\frac{1}{N} \sum_{n=1}^{N} a_n}$ (4.36)
33	Power	$P = \frac{1}{N} \sum_{n=1}^{N} a_n^2$ (4.37)
34	Sharpness	$Sharp = .11 \cdot \frac{\sum_{n=1}^{N} n \cdot g(n) \cdot a_n^{.23}}{N}$ $g(n) = \begin{Bmatrix} 1 & n < 15 \\ .066e^{.171n} & n \geq 15 \end{Bmatrix}$ (4.38)

Table 4.5: Other Spectrum Features (Cont.)

35	Slope	$slope = \frac{1}{\sum_{n=1}^{N} a_n} \cdot \frac{N \sum_{n=1}^{N} f_n a_n - \sum_{n=1}^{N} f_n \cdot \sum_{n=1}^{N} a_n}{N \sum_{n=0}^{N} f_n^2 - (\sum_{n=0}^{N} f_n)^2}$ (4.39)
36	Tristimulus 1	$Trist_1 = \frac{h_1}{\sum_{n=1}^{N} a_n}$ $h_k = \sum_n a_n \forall n \in \{N \mid k - .5 < \frac{f_n}{f_0} < k + .5\}$ (4.40)
37	Tristimulus 2	$Trist_2 = \frac{h_2 + h_3 + h_4}{\sum_{n=1}^{N} a_n}$ (4.41)
38	Tristimulus 3	$Trist_3 = \frac{\sum_{n=5}^{N} h_n}{\sum_{n=1}^{N} a_n}$ (4.42)
39	Inharmonicity	$inharm = \frac{2}{f_0} \frac{\sum_{n=1}^{N} \lvert f_n - h(f_n) f_0 \rvert \cdot a_n^2}{\sum_{n=1}^{N} a_n^2}$ $h(f) = floor(\frac{f}{f_0} + .5)$ (4.43)
40	Odd to Even Ratio	$OER = \frac{\sum_{n<H \forall n \in \{2k+1 \mid k \in N\}} h_n}{\sum_{n<H \forall n \in \{2k \mid k \in N\}} h_n}$ (4.44)
41	Signal Entropy	$H_s = -\sum_{n=1}^{N} P(a_n) log_2 P(a_n)$ (4.45)

greater to the total (4.41) and (4.42). The inharmonicity estimates the deviation from a purely harmonic signal. It calculates the ratio of energy not contained in a strict harmonic increments (4.43). The odd-to-even harmonic ratio gives the ration of power in the odd harmonics to the even harmonics (4.44). Finally, the spectrum entropy is the Shannon entropy contained in the amplitude distribution of the spectrum (4.45).

$$A_k = \frac{1}{(N-k)\sigma^2} \sum_{n=1}^{N-k} (x_n - \mu)(x_{n+k} - \mu) \forall k < N \tag{4.46}$$

$$AMDF_k = \frac{1}{N} \sum_{n=0}^{N-k-1} (x[n] - x[n+k])^2 \forall k < N \tag{4.47}$$

$$AMDF_k = \frac{1}{N} \sum_{n=0}^{N-k-1} x[n] - x[n+k] \forall k < N \tag{4.48}$$

We also extract three common vector features from the signal windows which measures various statistics of the spectrum with a shifted version of the spectrum. These include the Autocorrelation, which measures cross-correlation of the signal with itself (4.46), the average squared difference function (ASDF) which measures the average squared distance of a signal with a shifted version of itself and the average magnitude difference function (AMDF) which measures the average difference of a signal with a shifted version of itself. For each vector quantity, we calculate the first 11 signal statistics presented in Table 4.3. This gives 11 features for the Autocorrelation (features 42-52), the ASDF (features 53-63) and the AMDF (features 64-74) giving a total of 74 statistical features extracted from each sensor per window of raw sensor data.

4.4 Conclusions

This chapter has presented the methodology behind our sample data set. We have a vast data set collected from two clinical trials and continuous laboratory testing. This data set has been hand labeled with twelve targets split between walking and non-walking and laps versus free yielding 30 distinct target data sets. We collect every possible stream of raw sensor data from our

phones. Additionally, we transform the coordinate systems of acceleration for forward walking and maximum variance and calculate the walking direction and direction of maximum variance during walking giving a total of 31 continuous sensor streams. Finally, we split the sensor streams into discreet 512 sample windows with 256 sample overlap and present a comprehensive list of 74 statistical features we extract from each sensor. This provides a total of 2,294 sensor features providing an exhaustive, inclusive list of features to train machine learning algorithms in the subsequent chapters.

CHAPTER 5

IDENTIFYING IMPORTANT INPUT FEATURES

Our analysis pipeline allows us to investigate 31 sensor streams. For each sensor stream, we extract 74 statistical features. This gives a total of $31 * 74 = 2,294$ continuous sets of data collected throughout the monitoring periods. Investigating the correlation between 2,294 input variables and target outcomes directly using machine learning methods requires that the set of inputs be reduced to the important or most correlated sensor stream and statistic feature pairs. This chapter presents an analysis of input feature reduction. The goal is to identify which sensor and statistic pairs have high correlation with a given target inference, which pairs have no correlation and should be discarded and which streams could potentially provide information and would therefore require obfuscation to maintain privacy.

Reducing the input variable space before applying machine learning is a well-studied topic [74]. In general, methods can be divided into three broad categories. Filter methods seek to identify the input vectors independent of the machine learning algorithm used. These methods often use an estimate of covariance and mutual information among variables to identify the variables most useful in prediction. Wrapper methods use a trained machine learning algorithm as a black box to build models and select the input variables which yield the highest accuracy score. Finally, embedded models are specific to a certain subset of machine learning models which can score the input vectors during iterative learning. Filter and embedded methods are generally faster than wrapper methods when calculating with a large set of input variables. Wrapper methods are slower because many iterations of training must be conducted for large sets of possible features. However, since wrapper methods use the same machine learning algorithm to score the features, they generally give the most accurate determination of the feature's value.

In addition to determining the top input feature vectors, we must determine which features contribute meaningful information to prediction re-

quiring protection to maintain privacy. This problem is notably different from the traditional problem of determining the *BEST* features since the set of *BEST* features may exclude related highly correlated redundant features which would not necessarily be most predictive. For example, if two features A and B are highly correlated, the list of *BEST* features will choose the feature with the highest predictability, A while discarding B as a redundant feature. However, when protecting privacy it is important to consider the predictive ability of B since obfuscating A may not diminish the predictive ability if an adversary instead uses B. We therefore develop methods to determine what subsets of input features allow accurate prediction. We will see that methods generally have a design tradeoff between scalability limited by runtime and accuracy. The final output from our analysis will be a list of highly correlated subsets of features and a list of safe features which when used in predictive models do not allow prediction accuracies above a user selectable privacy threshold calibrated relative to the random noise threshold.

5.1 Selecting the Top Features

We evaluate two broad methods to select top input features used to train our models. We conduct filtering of input variables using the FEAST toolbox, a popular framework which implements a number of filter methods for feature selection [75, 76]. We also implement a sequential forward selection method (SFSM) using both a k-means classifier and a support vector machine/regression as the internal statistical method to score the features. To score the features, we use the $F1$-score for the classifiers and the lowest mean absolute error score (MAE) for the regressions.

$$H(X) = -\sum_{x \in X} p(x) \log p(x) \tag{5.1}$$

$$H(X|Y) = -\sum_{y \in Y} p(y) \sum_{x \in X} p(x|y) \log(p(x|y)) \tag{5.2}$$

$$\begin{aligned} I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \end{aligned} \tag{5.3}$$

$$I(X;Y|Z) = \sum_{z \in Z} p(z) \sum_{x \in X} \sum_{y \in Y} p(x,y|z) \log \left(\frac{p(x,y|z)}{p(x|z)p(y|z)} \right) \\ = H(X|Z) - H(X|Y,Z) \tag{5.4}$$

$$norm(I(X;Y)) = \frac{I(X;Y)}{\sqrt{H(X)H(Y))}} \tag{5.5}$$

Various information theory metrics are useful in evaluating feature selection routines. Central to information theory is the concept of entropy. The entropy H of the distribution X, formally given in (5.1), is determined by the sum of the discrete probabilities of each value of X. The more uncertainty that X is in a distinct state, the higher the value of the entropy H. As with most probability functions, we can also condition the entropy as shown in (5.2). Intuitively, this formula gives the amount of uncertainty in X given that we know Y. The mutual information score given in (5.3) is a measurement of the amount of information that each individual distribution reveals about the other. Its definition can be manipulated in terms of entropy as seen in the right side of the given equation. Intuitively, this states that the mutual information score gives the amount of uncertainty in Y minus the amount of uncertainty in Y that is known if we know X. Thus, we are left with a measurement of the amount of uncertainty in Y if we do not know X. We can condition the mutual information as shown in (5.4). This equation represents the amount of mutual information between X and Y given we know Z. For convenience, we can normalize this metric into a range from 0 to 1 by dividing by the square root of the product of the entropy in each distribution as shown in (5.5). Thus, a score of 1 means that we know all of the unknown from X if we have Y or perfect mutual information and a score of 0 indicates X tells us nothing about Y.

Machine learning models are evaluated by a standard set of metrics. We use the $F1$-score to evaluate the accuracy of classification machine learning models as shown in (5.8). The $F1$-score provides a scoring of the accuracy of the classifier from a scale of zero to one. The score takes into account the precision (5.6) which measures the percentage of correct, true positive, classifications out of all positive classifications. The score also takes into account the recall (5.7) which measures the percentage of positive classifi-

cations out of all the classification that should be positive. The $F1$-score is basically the weighted average of these two values with equal weight to each value. In multi-label classifications we use the average $F1$-score over all binary one-versus-one classification scores. Regression models use two standard measures of error. The first is the mean average error (5.9). Given a prediction $\widehat{y}$ and a known correct value y, the mean absolute error is the average absolute difference between the two values. The mean squared error is simply the average squared difference between $\widehat{y}$ and y (5.10). We prefer the mean absolute error as the evaluation metric in our experiments.

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \tag{5.6}$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{5.7}$$

$$F_1 - score = 2 * \frac{precision * recall}{precision + recall} \tag{5.8}$$

$$MAE = \frac{1}{N}\sum_{n=0}^{N-1} |y_n - \widehat{y_n}| \tag{5.9}$$

$$MSE = \frac{1}{N}\sum_{n=0}^{N-1} (y_n - \widehat{y_n})^2 \tag{5.10}$$

5.1.1 Filter Methods

Filter methods rank the input feature's ability to classify a target independently of the machine learning algorithm used. These methods use information theory to estimate the amount of information in each input vector compared to the prediction target. We use the FEAST toolbox to test eleven popular filter techniques as outlined in the following sections [75, 76]. Since the techniques from information theory produce relatively simple formulas to estimate the relevance, the filtering is generally useful in situations where computational resources are limited or the input vector is too large to conduct wrapper or embedded methods. However, the empirical estimate of mutual information is only accurate if the number of samples is large enough to estimate the underlying distribution. If the number of samples does not

accurately reflect the underlying distribution, the estimate can be wrong causing error in the filter methods.

We now evaluate eleven feature selection techniques which have been proposed in previous work and have been recently evaluated for accuracy [76]. Each method attempts to solve the problem of scoring the inclusion of a given feature X_C from the total set of features X. The scoring in each method defines a score based on the selection of X_C taking into account the rest of the currently selected features S to maximize the notion of information about the target distribution Y. The filtering methods themselves attempt to optimize the feature selection by trading off three primary pieces of information. First, they look at the mutual information between the chosen input vector and the target of the classification $I(X_C; Y)$. Maximizing this value intuitively gives a set of inputs X_C which contains all the information in Y. To reduce redundant variables, many methods introduce a penalty for mutual information between chosen features and remaining features which can be represented as $I(X_C; X_s) \forall s \in S$. Thus, the methods would seek to minimize redundant terms. The last term attempts to limit the penalty for redundant terms by adding to the score if the redundancy is only apparent given knowledge of the target distribution Y represented as $I(X_C; X_s|Y) \forall s \in S$. Intuitively, if the redundancy is only apparent given the target distribution Y, then the mutual information between the input variables may still be relevant when predicting Y.

$$MIM(X_C) = I(X_C; Y) \tag{5.11}$$

$$MIFS(X_C) = I(X_C; Y) - \beta \sum_{s \in S} I(X_C; X_s) \tag{5.12}$$

$$mRMR(X_C) = I(X_C; Y) - \frac{1}{|S|} \sum_{s \in S} I(X_C; X_s) \tag{5.13}$$

$$CONDRED(X_C) = I(X_C; Y) + \gamma \sum_{s \in S} I(X_C; X_s|Y) \tag{5.14}$$

$$CMIFS(X_C) = \sum_{s \in S} I(X_C Y | X_s) \approx I(X_C; Y) - \sum_{s \in S} I(X_C; X_S) + \sum_{s \in S} I(X_C; X_S | Y) \tag{5.15}$$

$$JMI(X_C) = \sum_{s \in S} I(X_C; X_s; Y) = I(X_C; Y) - \frac{1}{|S|} \sum_{s \in S} I(X_C; X_s) - I(X_C; X_s | Y) \tag{5.16}$$

$$BG(X_C) = I(X_C; Y) - \beta \sum_{s \in S} I(X_C; X_s) + \gamma \sum_{s \in S} I(X_C; X_s | Y) \tag{5.17}$$

The first seven filtering methods use linear weights among the three metrics. The simplest filtering method is the Mutual Information Maximization (MIM) [77] which scores the input feature using the mutual information between the input feature and the target distribution (5.11). The Mutual Information Feature Selection (MIFS) [78] introduces a penalty term for mutual information between the current feature and list of selected features with the penalty controllable by adjusting the β weighting value (5.12). Max-Relevance Min-Redundancy (mRMR) [79] weights the penalty term by the inverse of the size of the selected feature set (5.13). The conditional redundancy method introduces a term to maximize the conditional mutual information between the current input vector and list of selected input vectors with the weight of the term being controlled by the constant γ (5.14). The Conditional Mutual Info Feature Selection (CMIFS) [80] method maximizes the score of the current input feature by mutual information given the set of already selected features. Interestingly, this method implements a combination of the mutual information, the penalty term and the conditional redundancy terms plus a few constants which do not affect the final ordering of features (5.15). The Joint Mutual Information (JMI) [81] method also scores based on the mutual information with a penalty term for mutual information while dismissing the effect of the conditional mutual information. Similar to the mRMR method, it weights the penalty term based on the inverse of the size of the number of selected input features (5.16). Fi-

nally, the beta gamma method scores the input feature by maximizing the mutual information, penalizing the mutual information and maximizing the conditional mutual information. The weighting of the penalty for redundant information can be adjusted by varying β and the weighting of the scoring of the conditional mutual information can be controlled by varying γ (5.17).

$$\begin{aligned} CMIM(X_C) &= min_{s\in S}[I(X_C;Y|X_s)] \\ &= I(X_C;Y) - max_{s\in S}[I(X_C;X_s) - I(X_C;X_s|Y)] \end{aligned} \tag{5.18}$$

$$IF(X_C) = min_{s\in S}[I(X_C X_s;Y) - I(X_s;Y)] \tag{5.19}$$

$$ICAP(X_C) = I(X_C;Y) - \sum_{s\in S} max[0, \{I(X_C;X_s) - I(X_C;X_s|Y)\}] \tag{5.20}$$

$$DISR(X_C) = \sum_{s\in S} \frac{I(X_C X_s;Y)}{H(X_C X_s Y)} \tag{5.21}$$

Additionally, we evaluate four methods which introduce non-linear weightings into their scoring criteria. While this makes the intuitive sense of the methods less straightforward, we include them for completeness of testing. The Conditional Mutual Information Maximization (CMIM) [82] method scores using the mutual information between the current feature and the target distribution with the penalty being the maximum of the difference between the mutual information and conditional mutual information of each of the previously selected features (5.18). The Informative Fragments method (IF) [83] considers the minimum over all previously selected features of the difference between the mutual information of the whole set of selected features with and without the currently scored feature included (5.19). Intuitively, this will select the feature with the largest gain in mutual information over the previous set of selected features. The interaction capping (ICAP) [84] method is similar to JMI and CMIFS but caps the contribution of the penalty term to positive values (5.20). Finally, the Double Input Symmetrical Relevance (DISR) [85] method normalizes the conditional mutual information by the total entropy in an attempt to offset any inherent bias toward rare features (5.21). All of these methods will be used to select input features to train and evaluate our machine learning models.

5.1.2 Wrapper Methods

Wrapper methods use a machine learning algorithm as a black box to determine the set of input features which yield the highest scoring model. We implement a sequential forward selection method [86] which begins by scoring every feature using a selected machine learning model. The highest scoring feature is then chosen and models are built with a combination of that feature and every remaining feature. This process repeats greedily choosing the input feature with the highest score each round. The algorithm continues for a maximum of ten rounds or until the score is not improved by selecting another input feature. We conduct two wrapper experiments with the first experiment using a k-means clustering algorithm to score features and the second experiment using a support vector machine or regression with a radial basis kernel. Each round, the feature with the highest mutual information score is chosen as the best feature for the k-means wrapper and the feature with the highest $F1$-score or lowest mean absolute error is chosen for the support vector classification or regression wrapper.

The first wrapper method uses a simple k-means classifier to identify input features which naturally form clusters with high correlation with a given prediction target. The k-means algorithm organizes data points into one of k cluster sets to minimize the mean distance between data points and cluster centers. K-means naturally identifies the ability of an input vector to distinguish between labels. Intuitively, if the input feature has distinct values for a given target, the clustering should identify a cluster for that target. The algorithm begins by assigning a set of k cluster centers. It then assigns each data point to the clusters to minimize the mean as shown in (5.22). The initial centers are then adjusted to the center of mass for each cluster. The algorithm repeats the assignment of points and cluster centering until a local minima is reached. Obviously, finding the global minima of (5.22) requires the correct initial choice of cluster centers. We use the kmeans++ algorithm shown to increase the efficiency of choosing the initial cluster centers [87]. The k-means algorithm is run for a minimum of 10 iterations per trial with a stopping condition of a delta of 0.1 for the inertia for each trial. The clustered data is then compared with the label ground truth using the normalized mutual information score. While limited to identifying distinct groups of similar input feature values, k-means clustering is fast and efficient allowing

quick investigation of all 2,294 input vectors for all 30 prediction targets by brute force giving us a first approximation of the correlation of each feature to each target.

$$\operatorname{argmin} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 \tag{5.22}$$

Given a set of n observations $(x_1, x_2, ..., x_n)$ cluster each observation into one of k sets $S_1, S_2, ..., S_k$ to minimize the average distance to the center of the clusters.

The second wrapper method uses a support vector machine (SVM) or support vector regression (SVR) with a radial basis kernel. The SVM classifier is introduced in detail in Chapter 6. The model used as a black box here uses the default hyper-parameters to increase running efficiency over the whole space of input vectors. For targets with discreet values, a support vector classifier is used trained with one-versus-one classifiers choosing the classification with the most votes. For target with a continuous distribution, a support vector regression is used to predict the value. The round scoring for the SVM is conducted by choosing the feature yielding the highest $F1$-score. The round scoring for the support vector regression uses the mean absolute error to score each round (5.9). The item with the lowest mean absolute error is chosen as the top feature.

5.1.3 Evaluating Feature Selection Methods

In total, we have 15 feature selection methods with 11 filter methods produced by the FEAST toolkit and 4 wrapper methods. We include the results of the full sequential forward analysis for the k-means and SVM called KMFS and SVMFS respectively. We also include the ordered list of first round scoring of all sensor features for the k-means and SVM called KM1 and SVM1 respectively. We are interested if the first round scoring is correlated enough to predict feature utility without requiring the full computationally expensive SFSM. This gives 15 lists of top features for evaluation.

Similarity among Feature Selection Techniques

Given 15 top feature selection techniques, it is natural to ask how similar the feature selection techniques choose top features and if it is necessary to evaluate all 15 methods simultaneously. Each filter selection routine returns an ordered list of most predictive features. Classical methods of ranking similarity of ordered lists are problematic since each list of top features contains a different subset of the entire list of 2,294 selectable features. We therefore use normalized discounted cumulative gain (NDCG) due to its ability to rank ordered lists when the two lists do not necessarily contain the same features. We define the ground truth to be an ordered list of N top features. We define a relevance ranking rel defining the scores $N, N-1, ..., 1$ for all N features in the list. Thus, the top feature has score N down to the last feature with a score of 1. Given a list to evaluate, the discounted cumulative gain scores the ordering of the new list as the relevance of the top feature in the list plus the sum of each subsequent relevance divided by the log of the position of the item in the list (5.23). Features in the evaluation list not present in the ground truth list are given relevance scores of zero. We then normalize the score by dividing the discounted cumulative gain by the ideal discounted cumulative gain. The ideal cumulative gain is the cumulative gains score attainable if the ideal list is perfectly ordered. Thus, we have a metric capable of evaluating feature ordering with a range of zero to one for ordered lists with differing sets of members.

$$DCG = rel_1 + \sum_{n=2}^{N} \frac{rel_n}{log_2(n)} \tag{5.23}$$

$$nDCG = \frac{DCG}{IDCG} \tag{5.24}$$

We calculate the top forty features using the filter methods and the first round of the SVMFS and KMFS wrapper methods over all 30 prediction targets. We consider a maximum of 10 features from the full SVMFS and KMFS noting that 16 of the targets terminate in fewer than 10 iterations. The remaining, features see little improvement from further features. We calculate the normalized discounted cumulative gain for each combination of features. Figure 5.1 plots a heat map of the scores between each top feature selection method. Of course, the scores between the same methods yield a

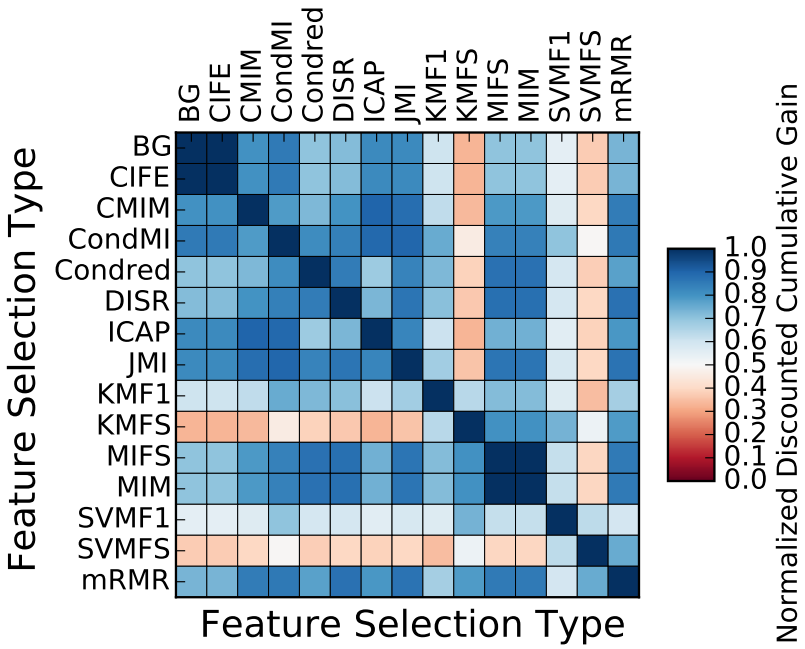

Figure 5.1: Normalized Discounted Cumulative Gain Scores between All Tested Feature Selection Techniques

perfect NDCG. The filter methods largely produce better orderings with less variation than both the KMFS and SVMFS which differ significantly from the orderings returned from the filtering methods. To better illustrate this, we look at the intersection of top features returned from the methods. For the top feature, every filter method returns the same number one feature. The filter methods only agree with the KMFS for 9.6% of the targets and agree with the SVMFS a mere 6.5% of the targets. The first round SVMF1 and KMF1 agree 100% of the time. We extend our analysis to the top 10 features as shown in Figure 5.2. We notice that even among filter methods, the highest agreement rarely surpasses 50%. The agreement with the KMFS and SVMFS is abysmal with many targets containing a single target in common. Figure 5.3 shows both the NDCG and average proportion of shared features. We see that most selection methods contain a mere 10% of shared features with mRMR giving the top score of 26.9%. Of course, this is rarely the top feature as demonstrated by the NDCG which is higher for the SVMI score since it always returns the same top feature. Filter methods return different top features in different orders than the wrapper methods using SVMFS. This motivates the need to evaluate the accuracy and utility of each method

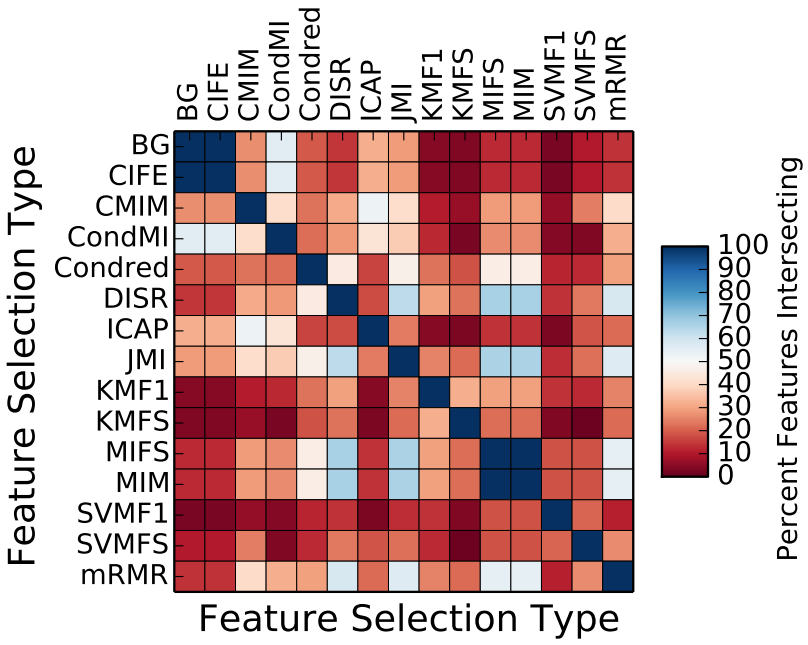

Figure 5.2: Intersection of Top Features between All Tested Feature Selection Techniques

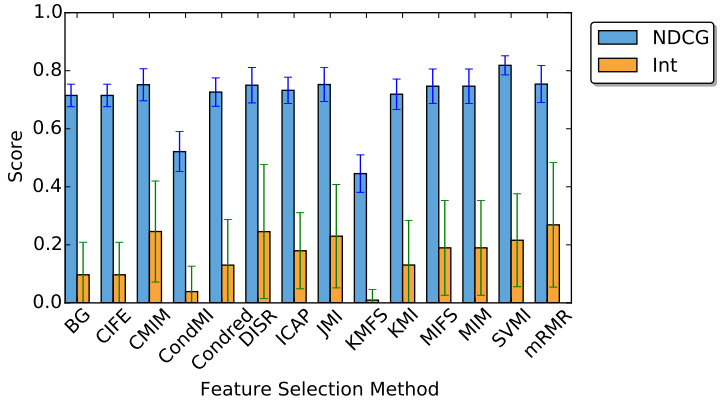

Figure 5.3: Normalized Discounted Cumulative Gain and Intersection of Top Features of All Feature Selection Compared to SVMF

individually. Thus, we continue evaluate all feature selection routines in the remainder of the dissertation.

Highly Correlated Features

We surmise a significant portion of the variation among top feature selection algorithms can be explained by the variation in rating highly correlated fea-

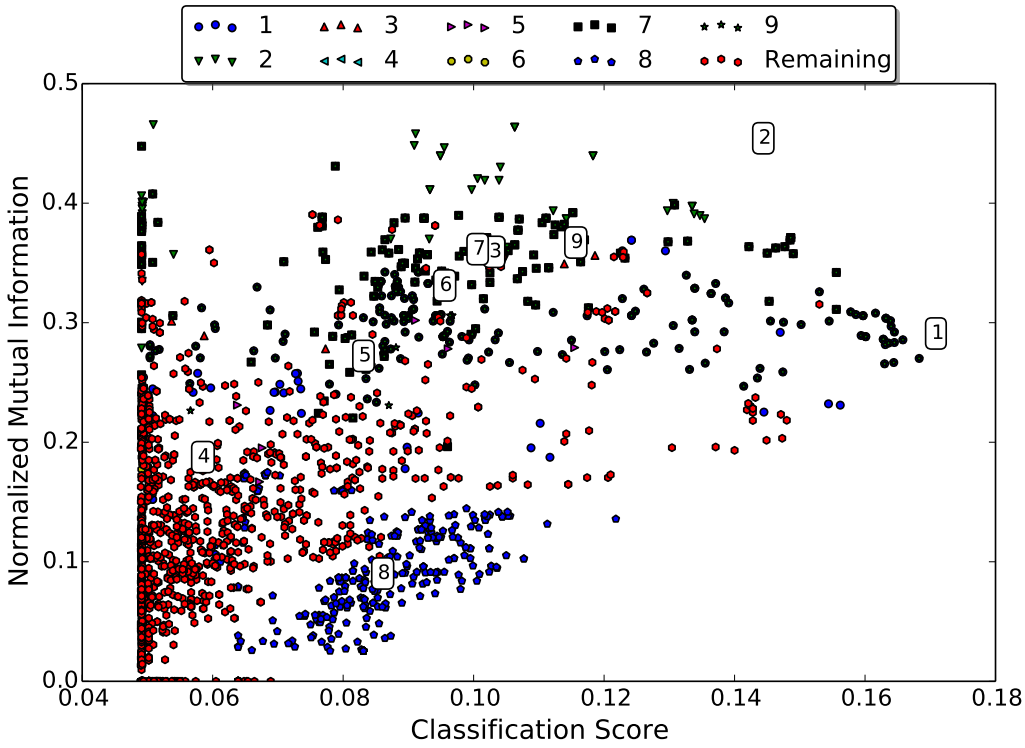

Figure 5.4: SVMFS First Round $F1$-score versus Normalized Mutual Information between the Feature and Target with Labels Indicating the Top SVMFS Features and Clustered Features Using Feature-Feature Normalized Mutual Information

tures. Consider two highly correlated, highly predictive features A and B. Feature selection methods are designed to choose what it deems the highest predictive feature (often determined as the feature with the highest normalized mutual information). Assuming it chooses feature A, most filter methods will then discard feature B. However, if feature B scores higher during the SVMFS, then it will be chosen even though it is discarded by the filter methods. We investigate how well the normalized mutual information calculated between feature vectors can cluster the feature scoring both in terms of normalized mutual information with the target of the classifier and with the SVMFS classification score shown in Figure 5.4. This figure plots every feature considered during the SVMFS routine. The top feature chosen by the classification score each round is labeled one through nine corresponding to the round the feature is selected as optimal. Each round, features with a NMIS greater than 0.5 with the optimally chosen feature are clustered into a per-round cluster. Each feature is then plotted by the NMIS with the tar-

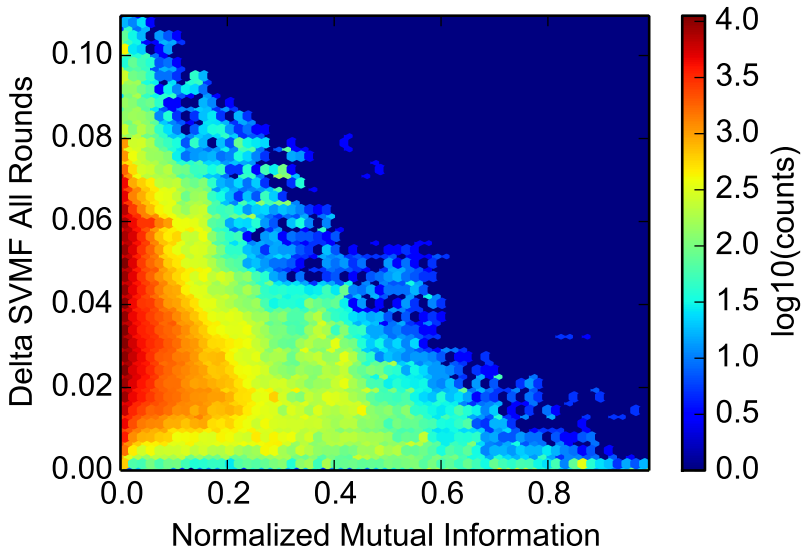

Figure 5.5: Feature NMI versus SVMFS Scores

get on the y-axis and the classification score improvement during the first round of the SVMFS on the x-axis. Interestingly, we see that the feature with the top classification score was not the feature with the highest normalized mutual information with the target. Thus, demonstrating why the filter methods return a different set of top features than the SVMFS method. We do see that features with high NMIS with other target features do seem to have similar classification scores and NMIS with the target. However, there are outliers in both cases.

Interestingly, we find that the optimal features chosen each round are rarely contained in an existing cluster from a previous round. We surmise this is due to correlated features containing redundant information. Thus, searching through the similar features unnecessarily wastes time in the sequential search. We therefore investigate the possibility of clustering features into similar groups using the normalized mutual information between features. Figure 5.5 shows the average correlation between the differences in SVMFS for all rounds of the SVMFS versus the NMIS of the features. We see that most features have little NMIS however for the ones that have high NMIS, the difference in scores during the SVMFS search is decreased. Thus, it seems that a high NMIS indicates a similar feature vector. It is also important to note that the converse is not necessarily true. A small difference in relative scoring between SVMFS rounds does not necessarily indicate a high NMIS.

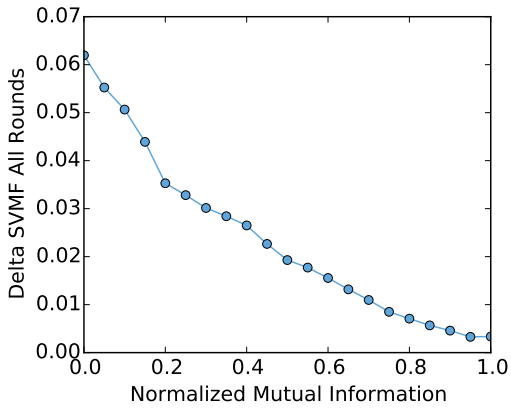

Figure 5.6: Average Feature NMI versus SVMFS Scores

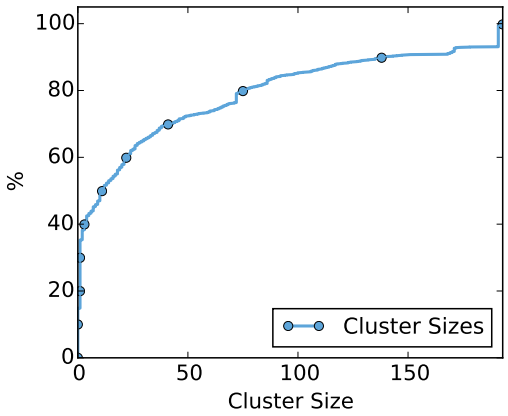

Figure 5.7: CDF of Cluster Sizes with a NMIS Cutoff of 0.5

Figure 5.6 plots the average difference between SVMFS scorings against the NMIS score averaged over all target features. We see the difference in delta SVMFS scores diminish as NMIS increases and choose 0.5 as the cutoff to consider features similar for clustering. Figure 5.7 demonstrates the cluster sizes formed by considering a NMIS cutoff of 0.5. We see that roughly 40% of features are independent of all others. The remaining features are contained in clusters with a large cluster at 77 and 179 features. These feature clusters can be reduced to a single feature thereby simplifying analysis during an SVMFS search.

We combine the clustered features and again compare the similarity of feature selection routines. Figure 5.8 shows the NDCG scores with the clustered features. We see overall improvement of scores. This is especially apparent in Figure 5.9 which plots the intersection of the top 10 features. While the intersection of the top 10 features had a maximum of 26.8% in the first experiment, using clustering, the maximum intersection increased to 60.6%. However, we still find that the feature selection methods contain enough different features to continue to investigate their top feature predictions. The analysis of highly correlated features presents the opportunity to reduce the set of features considered when classifying private features which will be explored in Section 5.2.

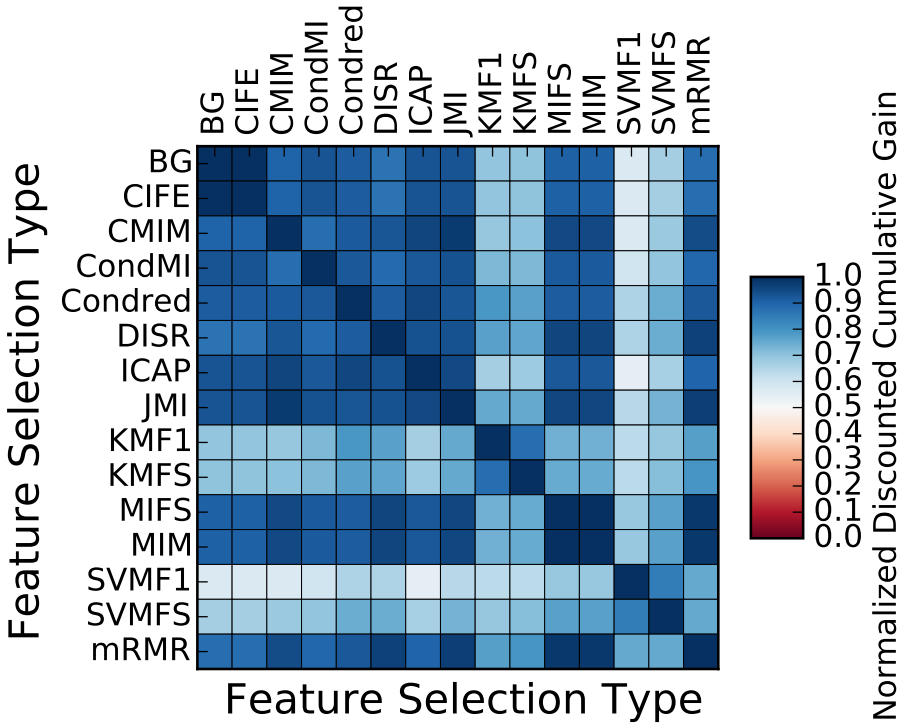

Figure 5.8: Normalized Discounted Cumulative Gain Scores between All Tested Feature Selection Techniques with Clustering

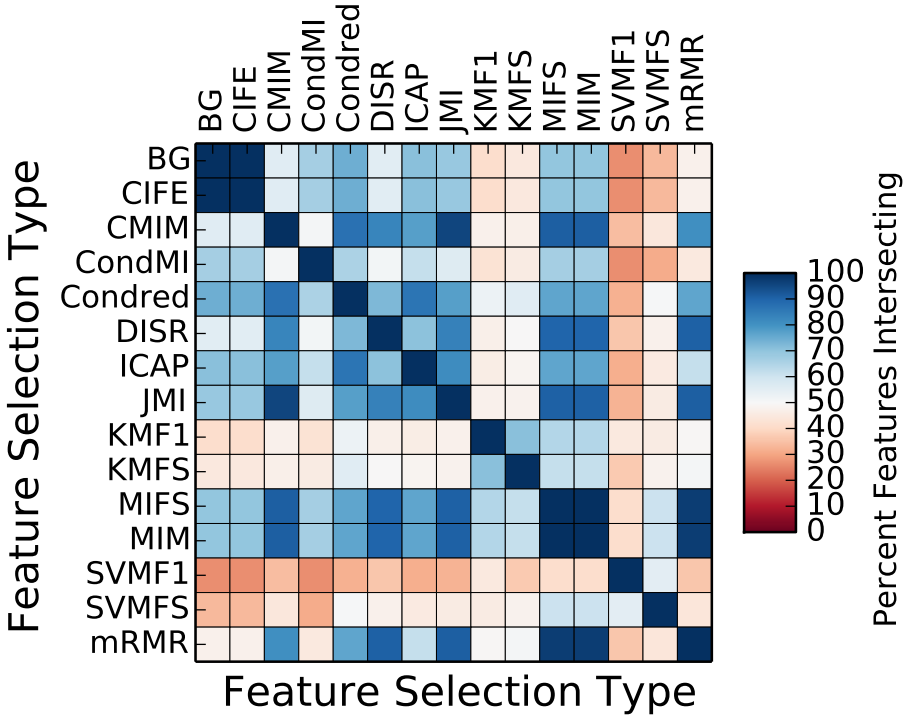

Figure 5.9: Intersection of Top Features between All Tested Feature Selection Techniques with Clustering

Table 5.1: Top Five Features Returned from Selected Feature Selection Methods

	SVMFS	mRMR	JMI	DISR
Activity	AccMag SpFlx AccX AMDFSkew GyroX SigKurt AccV SignMCR AccE SigKurt	AccMag SpFlx AccX ASDFSkew AccZ SpVar AccZ SpKurt AccV AcKurt	AccMag SpFlx AccZ AMDFSkew AccX SpFlx AccMag SigEnt AccMag AMDFMax	AccMag SpFlx dirF SigNZC AccMag SigEnt AccMag ASDFSkew AccMag SpEnt
FEV1/FVC W	GravZ SigMin AccZ SpRMS AccX SigSkew AccY SignMCR AccY SigMax	AccZ SigRMS AccX SigSkew AccMag SigCoV AccZ SigMin AccZ SigMax	AccZ SigRMS AccX SigSkew AccZ SigMax AccMag AmdfMean RotZ SigRMS	AccZ SigRMS AccX SigSkew RotZ ACMean RotZ SigMin AccMag SigSkew
phoneIDNW	GyroY SigMean AccMag SigRMS AccY SigSum RotZ SigMean AccMag AcSD	AccMag SigMean AccF SigMean AccP2 SpEnt AccV SpVar AccMag AcAvDev	AccMag SigMean RotZ SigMin AccX ACVar RotZ SigMax AccMag SigSum	AccMag SigMean AccX SigSum AccMag ACSD AccF SigMean AccMag SigSum
userIDW	AccSw AMDFMax AccZ SigMean AccMag SigSkew AccY SignMCR AccX ASDFSkew	RotZ SigMean AccMag AMDFMax AccZ SigRMS AccY SigSum AccZ SigMin	RotZ SigMean AccMag SigMin AccZ SigRMS AccX ACVar AccZ AMDFMean	RotZ SigMean AccY SigSum AccZ SigRMS AccMag SigEnt RotZ SigMin
speedW	AccMag AMDFMax GyroX AMDFMax AccMag SignMCR AccE SigAvDev AccZ AcSkew	AccZ SigRMS AccMag AMDFMean AccX SigMax AccZ SigMin AccMag SigSkew	AccZ SigRMS AccMag AMDFMean AccX SigMax AccZ SigMean AccMag SigSkew	AccZ SigRMS AccMag AMDFMean AccZ SigMin AccX SigMax AccZ SigMean

Top Ranked Features

Both of the forward selection methods give a base estimate of the accuracy of each classification. While the data is presented in this section, it is important to remember that the model's hyperparameters will be optimized in Chapter 6. However, the top selected features and relative scoring is presented here both for completeness and to motivate the need for privacy aware feature selection.

Table 5.1 shows the top selected features for the SVM forward selection, and the three filter methods with the highest feature intersection. As representative examples, we present the activity, FEV1/FVC while walking, phone identification when the user is not walking, user identification while the user is walking, and the walking speed of the user. We note wide variation in the features returned. As noted earlier, the filter methods tend to return similar features with the top feature often being identical. Unfortunately, these feature selection techniques differ greatly from the SVMFS method which is most likely the most accurate method to score the features. We finally note that most returned features are from the accelerometer, gyro-

scope and change in rotation all features we would expect to be important for classification and regression models.

We next present the relative classification accuracies of the KMFS and SVMFS. Each k-means model is fit to the set of input features with k being set to the number of target features. Regression models are fit by binning the target of the regression into ten distinct bins. The output from the trained model is then compared to the correct answers using the normalized mutual information scores. The silhouette score is also recorded. The silhouette score is measure of how distinct the clusters are in the model (5.25). Intuitively, more distinct clusters indicate a better model fitting to distinctly separable data. Table 5.2 shows the overall scoring of all classification and regression models. We see a range of NMIS scores with activity, speed, phone identification, and FEV1/FVC getting the highest scores. Unfortunately, we find these scores rather inaccurate as can be seen by the relatively low silhouette scores indicating the clustering is not creating unique clusters. This demonstrates the weakness of the k-means methods to score the features which can be overcome by higher-dimensional SVM models.

$$silhouette\ score = average\left(\frac{b-a}{max(a,b)}\right) \tag{5.25}$$

Here a is the distance from the point to its assigned cluster and b is the distance from the point and the nearest non-assigned cluster.

Table 5.3 presents the top $F1$-scores for each SVM classification model using a radial basis kernel. We see higher accuracy scores for activity classification closely followed by high scores for phone identification and COPD classification. We see lower scores for gender and user identification with little indication that users can be identified while stationary. Intuitively, this result makes sense since we expect the users to not be identified while stationary. Conversely, we expect phones to be identified with more accuracy through sensor fingerprinting while stationary. The mean absolute error rates for the regression models are given in Table 5.4. As expected, we can identify each characteristic with better accuracy while the users are walking. We can predict age within 10 years. The FEV1% seems difficult to predict directly from the raw features confirming the results from our previous work. The FEV1/FVC value, however, does have potential to be predictable. The

Table 5.2: Top Wrapper k-Means Cluster Scores

Target	NMIS	Silhouette Score
Activity	0.91	0.546
SpeedW	0.801	0.368
PhoneIDNW	0.777	0.896
FEV1/FVCW	0.744	0.456
UserIDW	0.736	0.333
PhoneIDW	0.719	1.0
PhoneIDAll	0.716	1.0
FEV1W	0.71	0.43
WeightW	0.701	0.427
FEV1/FVCAll	0.636	0.498
FEV1/FVCNW	0.623	0.476
AgeW	0.601	0.356
FEV1All	0.591	0.462
FEV1NW	0.573	0.619
UserIDAll	0.572	0.423
WeightAll	0.547	0.353
StepsW	0.547	0.296
WeightNW	0.544	0.481
CopdNW	0.54	0.472
UserIDNW	0.519	0.465
SpeedWLaps	0.453	0.393
AgeNW	0.451	0.561
AgeAll	0.445	0.278
HeightW	0.444	0.404
HeightNW	0.403	0.764
HeightAll	0.296	0.455
CopdAll	0.262	0.718
GenderW	0.225	0.919
CopdW	0.199	0.735
GenderAll	0.18	0.607
GenderNW	0.164	0.677

Table 5.3: Top Wrapper SVM Classification Scores

Target	Max $F1$-score
Activity	0.9948
PhoneIDNW	0.9402
PhoneIDW	0.9396
CopdW	0.9376
PhoneIDAll	0.9246
GenderW	0.9234
UserIDW	0.8814
CopdNW	0.8728
CopdAll	0.832
GenderNW	0.8157
GenderAll	0.7639
UserIDAll	0.4827
UserIDNW	0.459

Table 5.4: Top SVM Regression Scores

Target	Minimum MAE
AgeAll	10.5766
AgeNW	12.3722
AgeW	7.9577
FEV1All	15.0327
FEV1NW	17.1854
FEV1W	13.2044
FEV1/FVCAll	8.5551
FEV1/FVCNW	10.9753
FEV1/FVCW	7.3196
HeightAll	2.2185
HeightNW	2.0513
HeightW	1.7693
SpeedW	0.0574
SpeedWLaps	0.0671
StepsW	0.5812
WeightAll	25.3018
WeightNW	26.8351
WeightW	23.4036

speed and steps appear to be predictable with high accuracy. All regression and classification models will be optimized and scored in the Chapter 6.

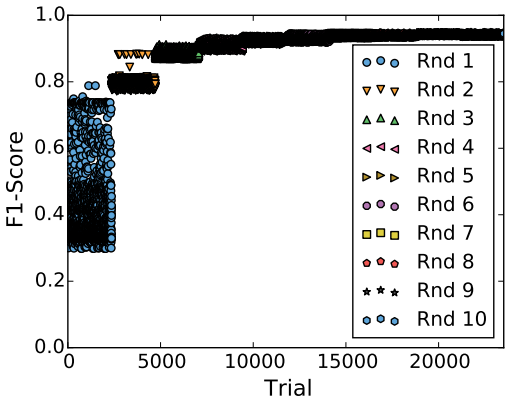

Figure 5.10: $F1$-score per Round of SVMFS

However, the prediction scores and wide variations in selected top features across feature selection methods demonstrate the need to further investigate which features are important to protect privacy.

5.2 Privacy Aware Feature Selection

Ideally, one of the top feature selection routines would return the set of top private features. However, we have seen that feature selection techniques provide wide variation in the top predicted features. To protect privacy, we must understand to what extent an individual feature can be used to leak information. This is particularly problematic since combinations of individual features can yield predictions not attainable from the separate features. Additionally, the top feature selection algorithms will discard features with high mutual information. A typical example of the score improvement per round for an SVM sequential forward selection routine is plotted in Figure 5.10. The selection of the first features establishes the highest improvement of $F1$-score with diminishing returns for each subsequent round. In the brute force case, we could simply evaluate every possible subset of feature combinations; however, this would quickly become impractical for large numbers of input features. In this section, we will develop techniques to classify features given a privacy score cutoff. We will also measure relative computation time and accuracy of various feature selection techniques to classify the privacy level of the features.

```
Result: Privacy Sensitive Features
searchFeatures = allFeatures
privateFeatures = []
topFeature, topScore = SFSMScore(searchFeatures)
while topScore > privThreshold do
    privateFeatures.add(topFeature)
    searchFeatures.remove(topFeature)
    topFeature, topScore = SFSMScore(searchFeatures)
end
return privateFeatures
```

Algorithm 1: Full Base Algorithm for Identifying Privacy Sensitive Features

5.2.1 Identifying Private Features

Algorithms to Search for Private Features

We develop three algorithms to identify the privacy sensitive features in our data. We want to evaluate the predictive capability of each input feature using an SVM or SVR model with radial basis kernel. The brute force method for feature selection would be to evaluate every possible combination of input features; however, this would result in the training and evaluation of 2^N machine learning models. Since the training of machine learning models is non-trivial for our large data set, we must optimize this procedure.

We first reduce the number of required models with our base algorithm by implementing a sequential forward search method at the core of our search as given in Algorithm 1. The full base algorithm begins with the set of all features. With each loop, it conducts a SFSM. If the feature with the top score is above the privacy threshold, it is removed and the SFSM is conducted again with the set of features minus the last optimal feature. While it is possible this method could miss features, SFSM has been shown to rank features with high accuracy. However, this algorithm still suffers from high computation time. On the full data set, we have seen it take up to two weeks to conduct a full SFSM search during our tests running on a single core of an AMD Opteron 8431 for a single target. This can be improved with parallelization, but for larger data sets, runtime must be improved.

We have seen that certain features can be clustered by computing the normalized mutual information between the feature vectors. Using these clusters, we can optimize the runtime of the algorithm by removing entire clusters of feature scores instead of single features after each round of the

```
Result: Privacy Sensitive Features
searchFeatures = allFeatures
privateFeatures = []
topFeature, topScore = SFSMScore(searchFeatures)
while topScore > privThreshold do
    privateFeatures.add(topFeature)
    searchFeatures.remove(topFeature)
    for each MIScore do
        if MIScore > 0.5 then
            privateFeatures.add(MIFeature)
            searchFeatures.remove(MIFeature)
        end
    end
    topFeature, topScore = SFSMScore(searchFeatures)
end
return privateFeatures
```

Algorithm 2: The MI Algorithm for Identifying Privacy Sensitive Features Optimized with Mutual Information

SFSM search. We implement this in our MI algorithm as given in Algorithm 2. The MI algorithm also searches for the top feature found each round of the SFSM search; however, upon identifying the top feature of interest, it also removes all features with high normalized mutual information scores to the identified optimal feature. This greatly decreases the runtime by decreasing searches depending on the cluster sizes of the top feature, but may introduce false positives if the features with high NMIS are not actually predictive of the target.

We consider Algorithm 1 as the baseline for accuracy when identifying private features. The MI algorithm, Algorithm 2, decreases runtime but may introduce false positive due to some features with high NMIS actually having a low correlation of predictability. We finally present the MIS algorithm given in Algorithm 3 which adds a second check before eliminating features with high NMIS. This algorithm also requires clustered features have less than a 0.01 difference in the first round of the SVMFS for the given target to be considered private. This should ensure that a feature not only has high NMIS, but also has similar predictive accuracy when predicting the target during the less computationally expensive first round of the SVMFS. We finally implement a shorter version of the full SVMFS search which terminates as soon as the maximum prediction score is surpassed during the SVMFS search lowering the number of iterations required during the SVMFS search for top features which are highly predictive.

```
Result: Privacy Sensitive Features
searchFeatures = allFeatures
privateFeatures = []
topFeature, topScore = SFSMScore(searchFeatures)
while topScore > privThreshold do
    privateFeatures.add(topFeature)
    searchFeatures.remove(topFeature)
    for each MIScore do
        if MIScore > 0.5 and DeltaSVMF1 < 0.01 then
            privateFeatures.add(MIFeature)
            searchFeatures.remove(MIFeature)
        end
    end
    topFeature, topScore = SFSMScore(searchFeatures)
end
return privateFeatures
```

Algorithm 3: The MIS Algorithm for Identifying Privacy Sensitive Features Optimized with Mutual Information and SVMF First Round Delta Scoring

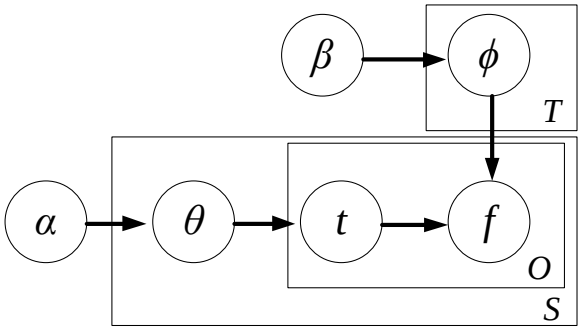

Figure 5.11: Latent Dirichlet Allocation Plate Diagram

Advanced Clustering with Latent Dirichlet Allocation

While k-means has classically been used to cluster individual samples into obvious groups, the method is simplistic. We investigate the ability of an advanced clustering method to predict the usefulness of a feature. Latent Dirichlet Allocation (LDA) was originally designed to classify the topics associated with words in a document. It is a generative model which given a set of observations attempts to establish unobserved clustering that explain the similarity seen across sessions. It has recently been found useful in natural language processing, genetics, and image processing. LDA models are trained through Gibbs sampling [88, 89]. We use the LDA implementation in Python to build the LDA models [90].

We adapt our problem to LDA by defining an individual testing period with a specific target as a session. Each sensor reading is discretized with the value being binned into a feature word encoded with the sensor, feature and binning value. The number of topics is set to the number of classification targets or the number of bins in the discretization of the regression targets. LDA takes as input the sessions, word vocabulary, and number of topics generating both a set of unique topics and probability that a specific word in the vocabulary indicates that a session should be classified as one of the identified topics. The plate representation seen in Figure 5.11 demonstrates how the observed feature words f is annotated for each observation $o \in O$. The hyperparameters α and β are priors for Dirichlet distributions $\theta \sim Dir(\alpha)$ and $\phi \sim Dir(\beta)$ representing distributions of topics per session S and topics over feature words respectively. For each session $s \in S$, a topic classification $t \in T$ is drawn from θ to define the topics assigned to the session with probability $\theta_t^{(s)} = P(t|S)$. Similarly, for feature word f, a topic $t \in T$ is sampled from ϕ and assigned to the feature word with probability $\phi_f^{(t)} = P(f|t)$. The LDA algorithm uses Gibb's sampling to estimate the topic assignment to features as shown in (5.26). Then, the assignment distributions can be trained by (5.27) and (5.28).

$$P(t_i = j|t_{-i,f}) \propto \frac{n_{-i,j}^{(f)} + \beta}{n_{-i,j}^{(\cdot)} + F\beta} \frac{n_{-i,j}^{(s_i)} + \alpha}{n_{-i}^{(s_i)} + T\alpha} \tag{5.26}$$

$$\theta_j^{(s)} = \frac{n_j^{(s)} + \alpha}{n_\cdot^{(s)} + T\alpha} \tag{5.27}$$

$$\phi_f^{(j)} = \frac{n_j^{(f)} + \beta}{n_j^{(\cdot)} + F\beta} \tag{5.28}$$

Since we set our number of topics as the number of unique targets, we would ideally like the algorithm to map an individual set of observations to an individual target. This is accomplished by choosing a small number for the α prior. We therefore set α to 0.001. We would like the number of feature words considered for each topic to be large in order to classify the relative importance of each feature word to the topic. Thus, we set β relatively large at 0.1. The LDA analysis provides us with a ranked probability of topics per session and ranked probability of topic probabilities per feature word.

The ranked probability of topic probabilities allows us to determine how important each feature word is to predict the topic thus giving us another way to rank features.

Classical Feature Selection Techniques

We investigate the relationship between the privacy of features and the normalized mutual information score with the target of the predictions. The normalized mutual information score is often the most widely accepted information metric when designing filter methods, thus we would expect the NMI score to approximate the privacy value of the features. We will also evaluate how well the 15 top feature selection routines eliminate the top private features as identified by our three algorithms including the 11 filter methods and four wrapper methods.

5.2.2 Private Feature Evaluation

Evaluating the ability of all methods to determine the private data set with the full set of sensors would take a significant amount of computation. To keep the experiments manageable, we limit our input features to the features extracted from the magnitude of acceleration, magnitude of the gyroscope and the orientation in the Z direction. We choose these sensors as a sampling of the three main types of sensors available to the models including a motion sensor, a rotational velocity sensor and an orientation sensor. We investigate five targets including the identity of the phone collecting the motion readings, the identity of the user, the activity, the FEV1/FVC of the subject, and the walking speed. This gives us a data set with 222 continuously collected features over 45 hours of readings tested on three classification targets and two regression targets.

Private Feature Algorithms

We first compare the results from the three private feature selection algorithms. Each algorithm is run five times with various privacy thresholds. For the classification, we use the $F1$-score to determine the privacy threshold. Thus, the algorithms iteratively remove features until the forward selection

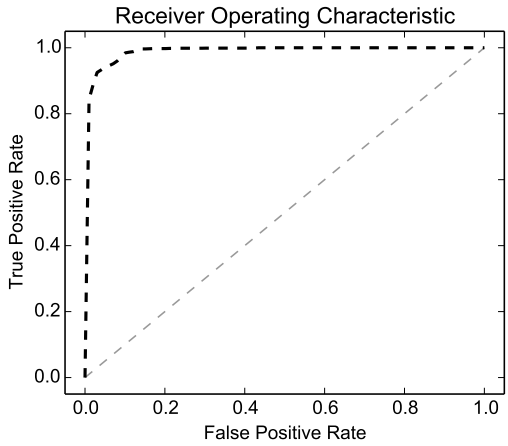

Figure 5.12: ROC Curve of SVM Phone Classification Using All Features from SVMFS (Area Under Curve = 0.99)

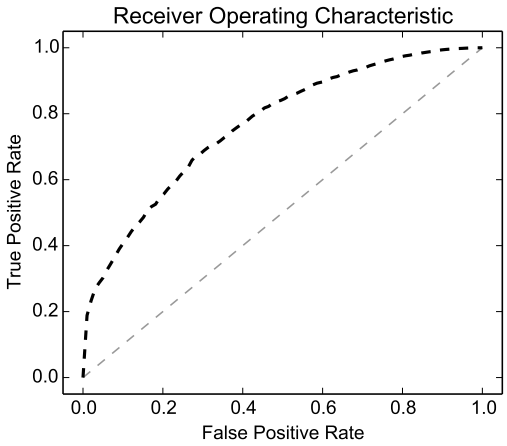

Figure 5.13: ROC Curve of SVM Phone Classification Using Safe Features from Privacy Algorithm #1 (0.7 Privacy Threshold) (Area Under Curve = 0.74)

cannot attain a $F1$-score higher than the threshold. We test at $F1$-score cutoffs of $0.4, 0.5, 0.6, 0.7$ and 0.8. Since the regression returns variable absolute errors depending on scale of the prediction target, we must define a privacy metric that is comparable across regressions. We first calculate the interval from the best classification score attained during the optimal forward search and the noise threshold or level of prediction when the regression algorithm is given purely random noise. This represents the interval between the best prediction and worst prediction. We define our privacy metric as the percentage from the optimal prediction to the noise level. We test the regression with privacy thresholds set at $0.2, 0.4, 0.6$, and 0.8 on this interval i.e. for 0.2 features are removed until the best MAE score is greater than 20% of the difference between the maximum MAE score and the noise threshold.

We consider our base algorithm, Algorithm 1, as the standard for identifying the private features. Figures 5.12 and 5.13 show the ROC curve for predicting the phoneID both with and without the identified sensitive features with a privacy level of 0.7. The figures clearly demonstrate that with all features, the phone classification attains high accuracy but with the feature elimination, the remaining features still cannot accurately predict the phone above roughly 70% combined accuracy. The three algorithms return slightly different sets of private features. Figure 5.14 gives the average num-

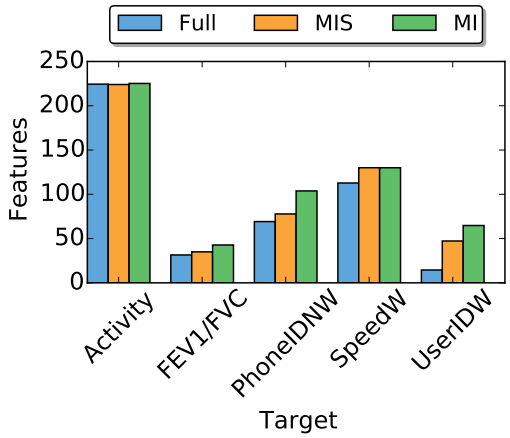

Figure 5.14: Private Features Identified Using Three Private Feature Identification Algorithms Averaged over All Privacy Levels

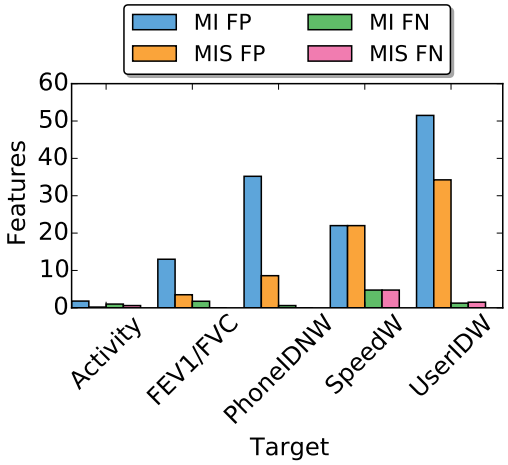

Figure 5.15: False Positive and False Negatives for MI and MIS Private Feature Selection Compared to the Full Algorithm Averaged over All Privacy Levels

ber of private features identified for each of the three algorithms over the five experiments. The MI and MIS algorithms returned more features than the full method with the straight MI returning more than the MIS for the phone identification, FEV1/FVC, and user identification. We found that the MI and MIS performed similarly for the activity and speed predictions. Overall, we see many more features identified as private for activity especially for lower privacy levels. This is hardly surprising since activity is a binary classifier and is the easiest classification for the motion sensor. We find that health and user identification contain the lowest number of private features in our sample data set. In most tests, the baseline returned the lowest number of private features followed by the MIS method and the MI methods. This is expected since the baseline method only eliminates one feature at a time, the MI method eliminates all clustered features, and the MIS method eliminates all clustered features that have a similar first round SVMFS score. Figure 5.15 shows the average false positives and false negatives compared to the baseline private feature identification algorithm. Since most features are identified as private for activity, we see a small number of false positives and negatives. For the rest of the targets, we see the MI algorithm returning the most false positives with the MIS algorithm correctly filtering a majority of false positive with both the FEV1/FVC capacity and the phone identifi-

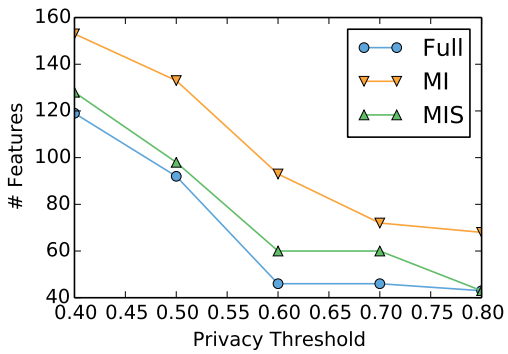

Figure 5.16: Phone Identification Privacy Sensitive Features versus Privacy Level (Lower = More Private)

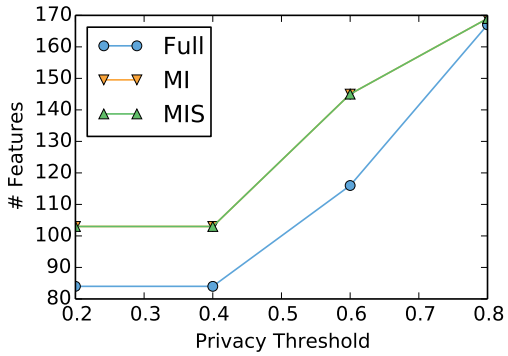

Figure 5.17: Walking Speed Privacy Sensitive Features versus Privacy Level (Higher = More Private)

cation. We see that the MIS is fairly successful at lowering the features with high NMIS but low similarity in predictive utility. There are relatively few false negatives for each method. For each false negative, we carefully retrain the classifier to identify the attainable accuracy compared to privacy threshold and find that the classification scores with the added false negative tests never surpass 0.01 above the target privacy threshold. The analysis indicates the clustering is most likely to assign false negatives to features which are close to the privacy threshold.

Figures 5.16 and 5.17 present the number of private features for the phones and walking speed over various privacy thresholds. The phones followed the ideal case with the MI yielding the highest number of features, followed by the MIS method with the base method yielding the smallest number of private features. As expected, the number of private features increases as the privacy threshold is lowered (becomes more strict) and the number of private features decreases as the privacy threshold is raised. For a privacy threshold of .6, 46 features are identified as private by the full privacy search with the MIS yielding 60 and MI algorithm yielding 93 features. Interestingly, the MI and MIS methods yields similar results for features. Thus, the MIS methods does not always eliminate the false positives from features with high NMIS. Once again, we see the number of private features rise as the privacy threshold is lowered. We surmise 0.4 as a reasonable level of privacy for the regression with 84 features being identified as private by the full algorithm and 103 being identified as private by the MI and MIS methods. Finally, Figures 5.18 and

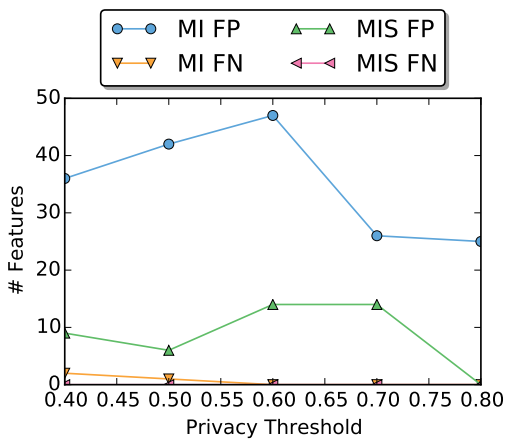

Figure 5.18: Phone Identification Sensitive Features Identification False Positives and Negatives (Lower = More Private)

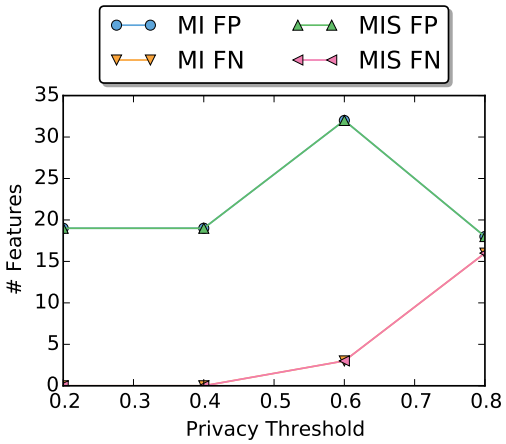

Figure 5.19: Walking Speed Sensitive Features Identification False Positives and Negatives (Higher = More Private)

5.19 illustrate the false positives and negatives for the phone identification and walking speeds at various privacy levels. We do not observe any patterns across the five experiments in false positives; however, we do note that the false negatives seem to be more prevalent at stricter privacy thresholds. We surmise that a privacy level which is too strict begins to be affected by more subtle combinations of less predictive features. As the classification score requirement decreases, it becomes more likely that combinations of features with little predictive utility can combine to edge over the threshold. This is encouraging since false negatives with low predictability will not substantially hurt the privacy of the user.

Tables 5.5 and 5.6 give the runtimes in hours of each of the private feature identification algorithms. It is important to note that all runtimes are conducted on a single core of an AMD Opteron 8431. The algorithms do conduct a large search of machine learning training which can be run in parallel; however, to compare the overall runtime, we limit our analysis to a single core. We find that higher privacy thresholds which identify more features tend to finish more quickly. Lower privacy thresholds have a tendency to have more features on the boundary thus lengthening the search. Regression searches are generally shorter since the underlying machine learning SVR model only needs to be trained once. The FEV1/FVC models with lower number of private features finished in as little as 36 minutes with the worst case only

Table 5.5: Privacy Feature Selection Runtime Comparison for Classification Targets

	Activity			PhoneIDNW		
Method	Base(h)	MI (h)	MIS (h)	Base(h)	MI (h)	MIS (h)
Best	.14	.08	.09	3.4	2.3	3.1
Average	15.06	7.3	9.0	6.4	2.7	4.0
Worst	44.9	21.8	26.7	8.2	3.2	4.6

	UserIDW		
Method	Base(h)	MI (h)	MIS (h)
Best	11.0	9.0	10.6
Average	142.3	35.0	44.1
Worst	165.5	55.3	62.7

Table 5.6: Privacy Feature Selection Runtime Comparison for Regression Targets

	FEV1/FVC			SpeedW		
Method	Base(h)	MI (h)	MIS (h)	Base(h)	MI (h)	MIS (h)
Best	.6	.6	.6	23.2	7.9	8.3
Average	6.3	1.8	3.4	32.3	9.9	9.6
Worst	13.8	3.3	7.0	45.0	12.8	11.6

reaching to 13.8 hours. Of course, regression trials with more private features can take almost 2 days for the base algorithm. We do see substantial speed improvements by using the MI and MIS algorithms. The classification takes much longer since the SVM classifier is composed of many one-versus-one classifiers. This makes the user identification with 88 targets particularly computationally expensive. With the base case, the privacy algorithm takes roughly two weeks to complete. The MI and MIS algorithms cut that down to 2-3 days. The number of private feature also increases the computation of the activity models. The shortest classification runtime was seen in the phone identification which has a relatively low number of sensitive features and targets compared the activity and user identification classifiers.

Overall, we see two primary effects which determine the total computation time. The number of classification targets greatly increases the complexity of an SVM classifier. We therefore recommend using a classifier which scales better with a high number of prediction targets. We will investigate other possible classifiers in Chapter 6. The number of features which yield pre-

diction rates close to the privacy threshold also increase the runtime since the forward selective search must search more rounds at the threshold. Since more features tend to be included with tighter privacy thresholds, a lower security threshold seems to increase the runtime. Overall, we see non-trivial performance gains for the MI and MIS algorithms with the MI performing the fastest followed by the MIS. Thus, we find a design trade-off between the accuracy of identifying the complete set of private features and computation.

Filter Method Top Features and Private Features

The private aware features selection routines presented in Section 5.2.1 quantify the relative privacy risk of features by conducting repeated forward selection searches using a machine learning model. This procedure is still computationally expensive requiring hours to days of computation. Ideally, filter methods could be used to select features since the methods carry far less computational overhead and therefore scale to higher number of features.

We look at the ability of the filter methods implemented in FEAST to predict the most sensitive features for various classification and regression targets. We conduct our analysis with a privacy threshold $F1$-score of 0.6. To compare all features, we conduct the FEAST search returning the ordered ranking of all 222 features. We then compare the top features returned from FEAST to the base private feature identification algorithm. If the FEAST feature is identified as a private feature, it is considered a true positive identification. If the FEAST feature is not a private feature, it is considered a false positive. Figures 5.20 and 5.21 plot the ROC for various filter methods identifying the private features for regression models including the FEV1/FVC and walking speed estimation respectively. Figures 5.22 and 5.23 show similar ROC curves for classification models including the phone and user identification models. The filter selection techniques perform similarly for the range of privacy thresholds tested with similar number of false positive and negatives for each particular classification target. We see that the top performing filter methods are the mutual information maximization, max-relavance min-redundancy, joint mutual information and double input symmetrical relevance methods. Both the conditional mutual information maximization and conditional redundancy perform similarly to random guessing with the remaining features actually performing worse than random guessing. Interest-

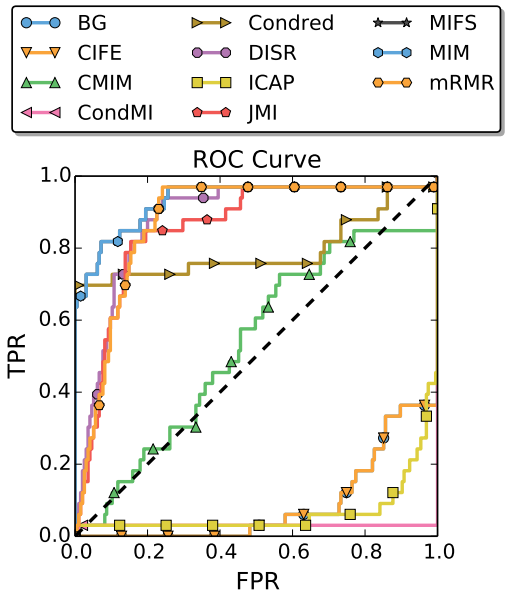

Figure 5.20: FEV1/FVC Filter Feature Selection Private Feature Identification ROC Curve (0.6 Privacy Threshold)

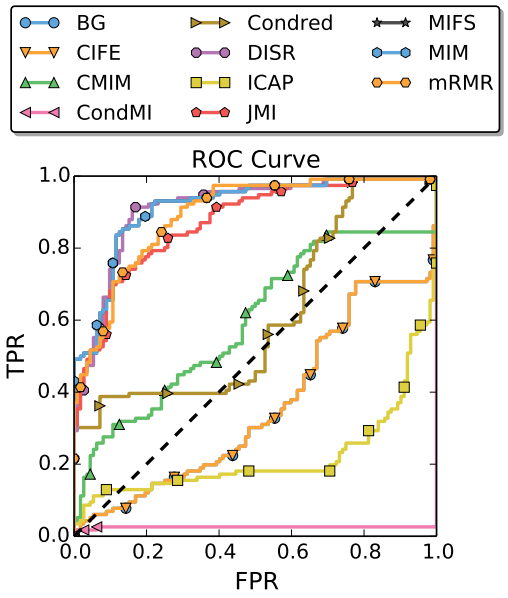

Figure 5.21: Walking Speed Filter Feature Selection Private Feature Identification ROC Curve (0.6 Privacy Threshold)

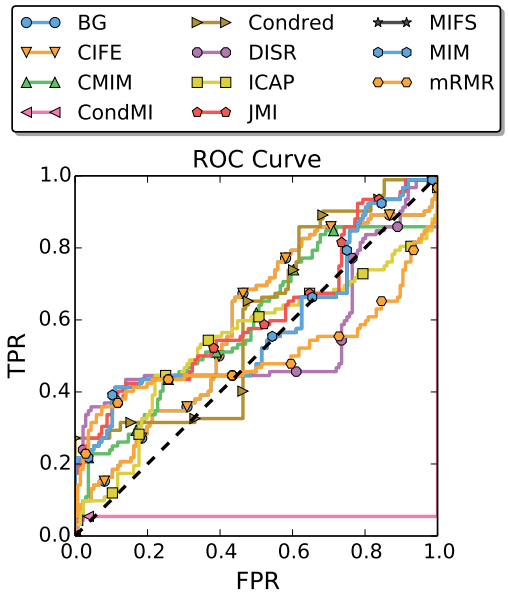

Figure 5.22: Phone Identification Filter Feature Selection Private Feature Identification ROC Curve (0.5 Privacy Threshold)

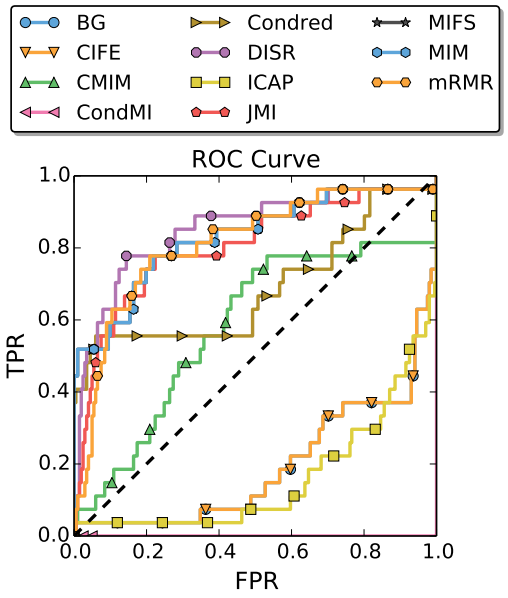

Figure 5.23: User Identification Filter Feature Selection Private Feature Identification ROC Curve (0.5 Privacy Threshold)

ingly, the same top features perform well for identifying the private features necessary to conduct the user identification classification. However, the filter methods perform quite poorly for the phone identification task with all methods tending to follow the diagonal best guess line.

Overall, we find that the filter selection techniques perform fairly well for the private features identified during regression analysis. The MIM method actually identifies 60% of the top private features before generating a false positive. All methods do generate false positives and negatives; however, they significantly reduce the computation time from days to mere minutes with total runtime of the FEAST algorithm taking 8 minutes per target feature. Unfortunately, the methods perform extremely poorly for phone identification tasks and reduced accuracy for user identification.

Other Metrics to Predict Private Features

Ideally, a low computation metric would be useful to identify private features eliminating the necessity of calculating an expensive feature elimination with forward selection search. We investigate three basic methods to classify private features including k-means clustering, NMIS with the target of the classifier, and a latent Dirichlet allocation feature scoring. The results are presented in Figures 5.24 and 5.25. We see that overall, the simple normalized mutual information score is most effective at identifying private features. The LDA scoring does an extremely poor job of classifying private features giving results similar to random in the best case and actually ordering features in the inverse order in the worst case. The k-means also does fairly well at classification actually beating the NMI for the phone ID. We find that using the k-means clustering and normalized mutual information with the target provide the best alternative to an expensive iterative search. While the search is guaranteed to provide ground truth, algorithms which require less computation are highly desirable for larger data sets with more possible features. We finally note that the results from the filter analysis agree with our recommendation to user the straight normalized mutual information. The top scoring filter method is actually the mutual information maximization function. Furthermore, the max-relavance min-redundancy, joint mutual information and double input symmetrical relevance methods all scale the penalty term by $\frac{1}{S}$ where S is the number of samples consid-

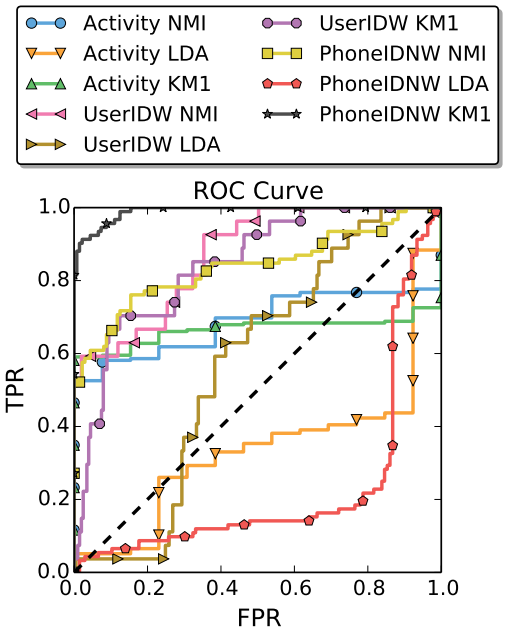

Figure 5.24: ROC Curve of NMI and LDA Rankings for Classification Private Feature Selection

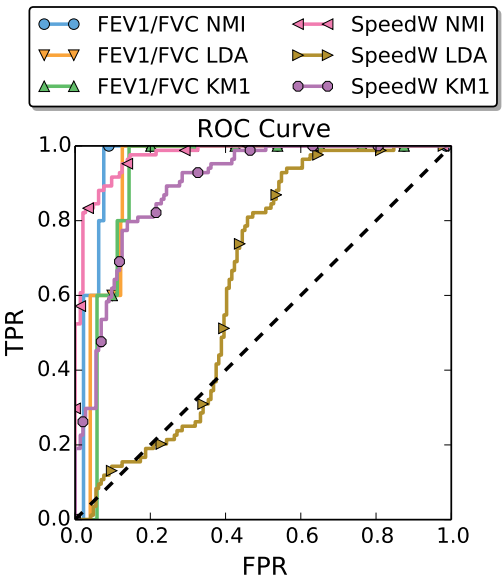

Figure 5.25: ROC Curve of NMI and LDA Rankings for Regression Private Feature Selection

ered. Since we are considering on the order of 10 k-1 M samples, this term is negligible when calculating the scores. Thus, the top scoring filter methods are simply using the normalized mutual information with the target of the model.

Overall, we find the basic algorithm for private feature identification returns the best estimate of the features important for the prediction models, but require high computation. Computation time can be reduced by considering eliminating groups of features which have high NMIS inside the group. However, this can lead to false positives. We reduce these false positives giving a hybrid MIS method by only grouping features with high NMIS and similar scoring during the first round of a forward sequential search using a SVM/SVR classifier. If computation is extremely limited, the best non-wrapper method to implement private feature selection is using the NMIS with the target directly. This is computationally efficient at the expense of a higher misclassification rate especially for models with many classification targets such as user identification.

5.2.3 Features Relevant to Privacy

We now identify the set of relevant private features out of the set of 2294 features contained in our full data set. We use the MIS algorithm outlined in the previous section. The algorithm searches for the set of private features by repeatedly searching for and removing the most predictive feature returning the set of safe features which cannot provide a classification score higher than the privacy threshold. The algorithm scores features using the highest $F1$-score for classification models and the lowest mean absolute error attained by removing the feature. We run for all target classification and regression targets returning an ordered list of important sensor features. We adaptively determine our privacy cutoff by calculating the point between the maximum prediction accuracy determined during the forward sequential search and the noise threshold. The sequential forward search is conducted using an SVM for classification or an SVR for regression both using a radial basis kernel. The noise threshold is determined by training the model with randomly generated noise and averaging the prediction scores over all available testing data.

We investigate what features appear useful and must be hidden to protect privacy. Table 5.7 lists the top 5 first identified private features per prediction target. We see that each prediction target returns different top private features. We break down the top features by sensor, features and sensor features and compare the intersection of top features between prediction targets in the subsequent sections.

Important Sensors

Figure 5.26 shows the percentage of sensors which were selected as top sensor features for each classification target. We see three primary classes of results for each target from the model. For targets with low prediction scores, the distribution of chosen sensors has a tendency to be more uniform than models with higher accuracy. This is unsurprising since models with lower accuracy will be more likely to over fit to sensors which may not be highly predictive. This is especially apparent in the COPDNW and FEV1NW which have a nearly uniform distribution of selected sensors. We see two primary types of useful information from the sensor linear motion and rotational motion. The FEV1%, FEV1/FVC and weight models have higher accuracy when training

Table 5.7: Top Five Private Features over All Prediction Targets

Activity	MagZ SigAvDev GyroZ AMDFAvDev AccZ SpHPS AccP1 ACAvDev AccMag SpFlx	AgeAll	AccY SpSum AccP1 AMDFAvDev AccF SigVar AccN AMDFVar AccY AMDFSD	AgeNW	GravMag SigMin MagMag ASDFNZC MagZ IrrJ HR SigMax RotX IrrJ
AgeW	AccV ASDFAvDev AccMag SpFlx AccV SpKurt GyroX ASDFSum AccN AMDFVar	CopdAll	GyroZ AMDFAvDev AccP1 ACAvDev MagZ IrrJ GyroX ASDFSum GyroY ASDFMax	CopdNW	GravY AMDFNZC MagZ IrrJ GravMag ACSkew GyroX SigMean GyroMag ASDFKurt
CopdW	GyroX IrrK GyroMag SigVar GyroZ AMDFAvDev GyroY SpSD GyroZ SigFF	FEV1All	GyroX AMDFMin GyroX IrrK GravMag SigMin GyroZ SigMin GyroMag SigVar	FEV1NW	AccSw ACSkew MagZ SigAvDev GyroZ AMDFAvDev AccP1 ACAvDev MagZ IrrJ
FEV1W	GyroX IrrK GyroMag AMDFSD GyroZ SigMin GyroMag SigVar GyroZ AMDFAvDev	FEV1/FVCAll	GravMag SigMin MagX AMDFAvDev MagZ AMDFMean AccMag SpFlx MagX ASDFAvDev	FEV1/FVCNW	AccX SpSlp RotX SigMax AccMag SpFlx AccF SignMCR AccY SpSum
FEV1/FVCW	MagX AMDFAvDev MagX ASDFAvDev AccY SpSum AccP1 AMDFAvDev AccF SigVar	GenderAll	DirOff SigMin GravY AMDFNZC POx SpSmooth AccMag SpFlx MagZ IrrJ	GenderNW	AccSw ACSkew DirOff SigMin DirF ACMax AccP1 ACAvDev AccMag SpFlx
GenderW	AccSw ACSkew DirOff SigMin GravY AMDFNZC GyroZ AMDFAvDev AccZ SpHPS	HeightAll	AccN SpRoll AccN SpSD GravMag SigNZC AccP1 SpSD AccZ SigMax	HeightNW	GravMag SigMin AccN SpSD GravMag SigNZC AccSw SpSmooth HR SigMax
HeightW	GravY AMDFNZC GravMag SigNZC AccMag SpFlx GravZ AMDFNZC AccV SpKurt	PhoneIDAll	GyroX IrrK GyroZ SigMin GyroMag SigVar GyroZ AMDFAvDev GyroY SpSD	PhoneIDNW	GyroZ AMDFAvDev AccP1 ACAvDev MagZ IrrJ GyroX ASDFSum GyroZ ACKurt
PhoneIDW	GyroZ AMDFAvDev GravX AMDFKurt RotY SpKurt AccN AMDFVar GyroX ASDFSum	SpeedW	AccV ASDFAvDev GravY AMDFNZC GyroZ AMDFAvDev AccZ SpHPS AccP1 ACAvDev	StepsW	MagZ SigAvDev AccV ASDFAvDev GravY AMDFNZC GyroZ AMDFAvDev AccZ SpHPS
UserIDAll	RotY SigMean AccY ACSum RotY ACVar AccY SigSum AccY ACSD	UserIDNW	AccP1 SigMean AccN SigSum AccF ACMean AccP1 ACAvDev AccP1 ACRMS	UserIDW	AccSw SpRMS AccP2 SpSkews AccV SigEnt AccZ SpEnt AccP2 SpSum
WeightAll	GyroX AMDFMin GyroX IrrK GyroZ SigMin GyroMag SigVar GyroZ AMDFAvDev	WeightNW	GyroZ AMDFAvDev AccP1 ACAvDev MagZ IrrJ HR SigMax GyroX ASDFSum	WeightW	GyroY SigSD GyroZ AMDFAvDev GyroY AMDFAvDev GyroMag SpCent GyroY SpEnt

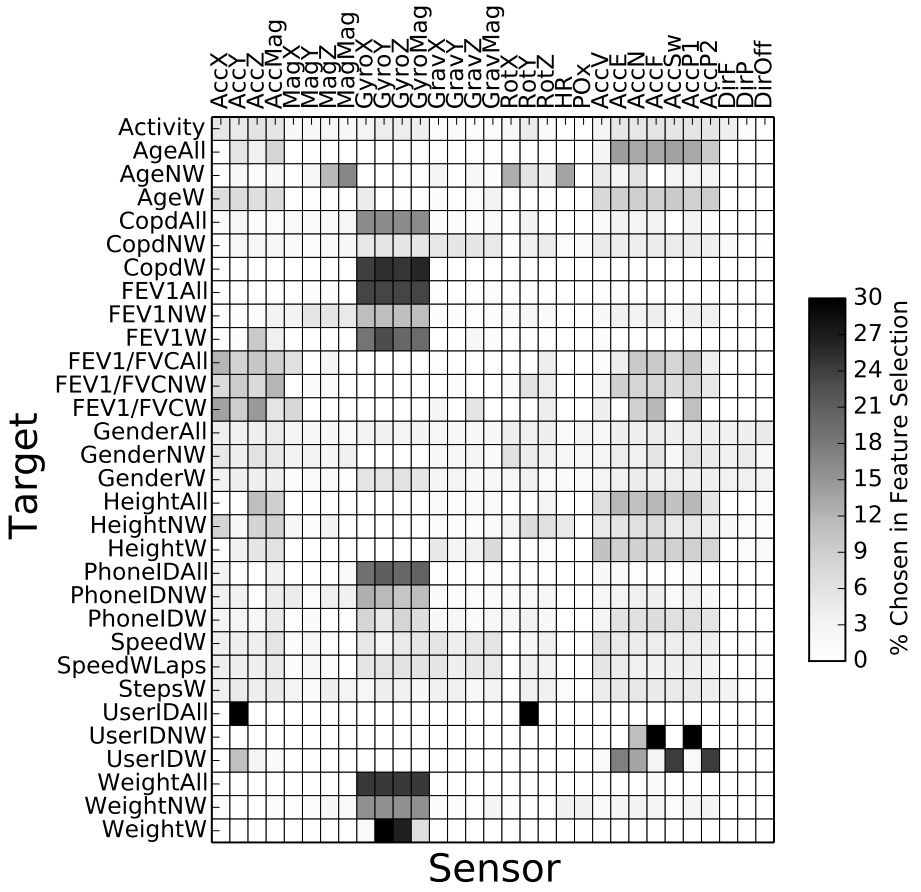

Figure 5.26: Private Feature Rates per Sensor

on sensors related to rotation of the device as measured by the gyroscope. Most other models attain higher accuracy looking at the linear motion sensors. We see models for speed and activity showing little preference for which reference frame used for the model. Interestingly, we see both user identification and age choosing acceleration streams in the walking reference frames. We finally note that the direction of walking and difference statistics of relative motion between forward and sway accelerations are almost never chosen as private features. Intuitively, the walking direction should not help predict these targets. Thus, not choosing the direction provides a nice sanity check for the methods. We note that these three streams can be safely eliminated for the remaining privacy analysis DirF, DirP and DirOff.

Important Statistical Features

Figure 5.27 shows the percentage of chosen statistical features across all sensors. We observe a wide variety of sensor features chosen as private features. Overall, prediction targets with low overall scores once again have a tendency toward a uniform distribution of statistical features. The important contribution of this analysis are what statistical features do not leak significant privacy leaks. The most noticeable features which contribute little to the privacy include the signal fundamental frequency, signal covariance, spectral minimum, spectral flatness, the three spectral tristimulus values, the spectral inharmicity, spectral odd/even ration, ASDF kurtosis, ASDF minimum, AMDF kurtosis and AMDF minimum. Identifying the features with little contribution to privacy allows us to further reduce our set of total sensor features from 31 * 74 = 2294 down to 28 sensors times 61 features for a total of 28 * 61 = 1708 thus eliminating 26% of the sensor features from consideration.

Important Sensor Statistical Features

Figure 5.28 lists the relative selection position of every sensor feature under consideration. We notice that the accelerometer and gyroscope sensors contribute greatly to most predictions. The gyroscope is particularly useful for predicting COPD, FEV1%, phone identification, and weight. Gender, a prediction with low classification accuracy, has the most uniform distribution of sensor features. Prediction targets including activity, speed and steps have higher prediction accuracy when using the accelerometer sensors. The diagram more clearly illustrates that the direction, heartrate, and pulse oximeter sensors give little predictive information. The rotation, magnetometer, and gravity sensors give less information than the gyroscope and accelerometer. However, as expected, many features give redundant information useful for prediction requiring a wide consideration for the necessary noise to maintain privacy in the raw data signal.

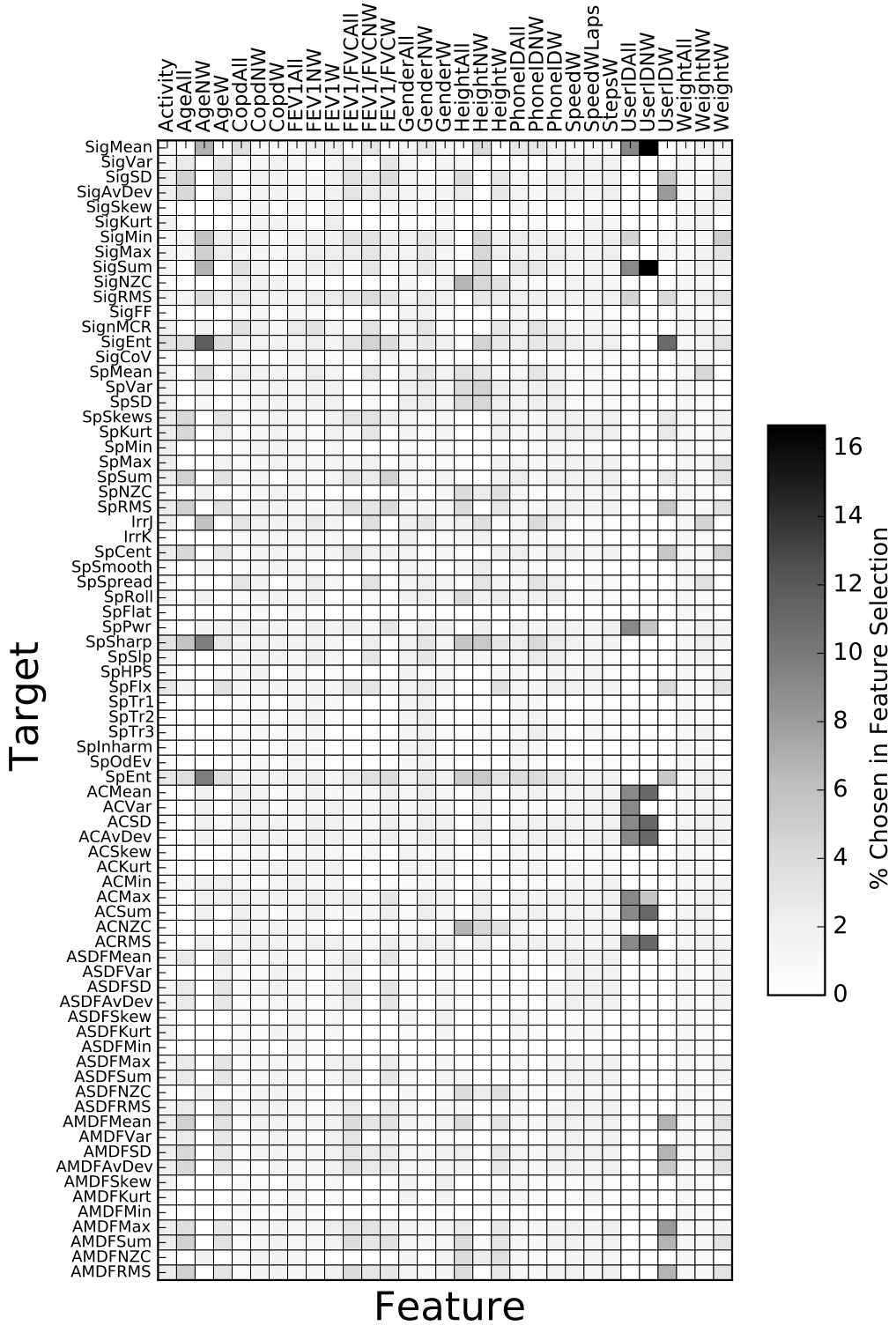

Figure 5.27: Private Feature Rates per Feature

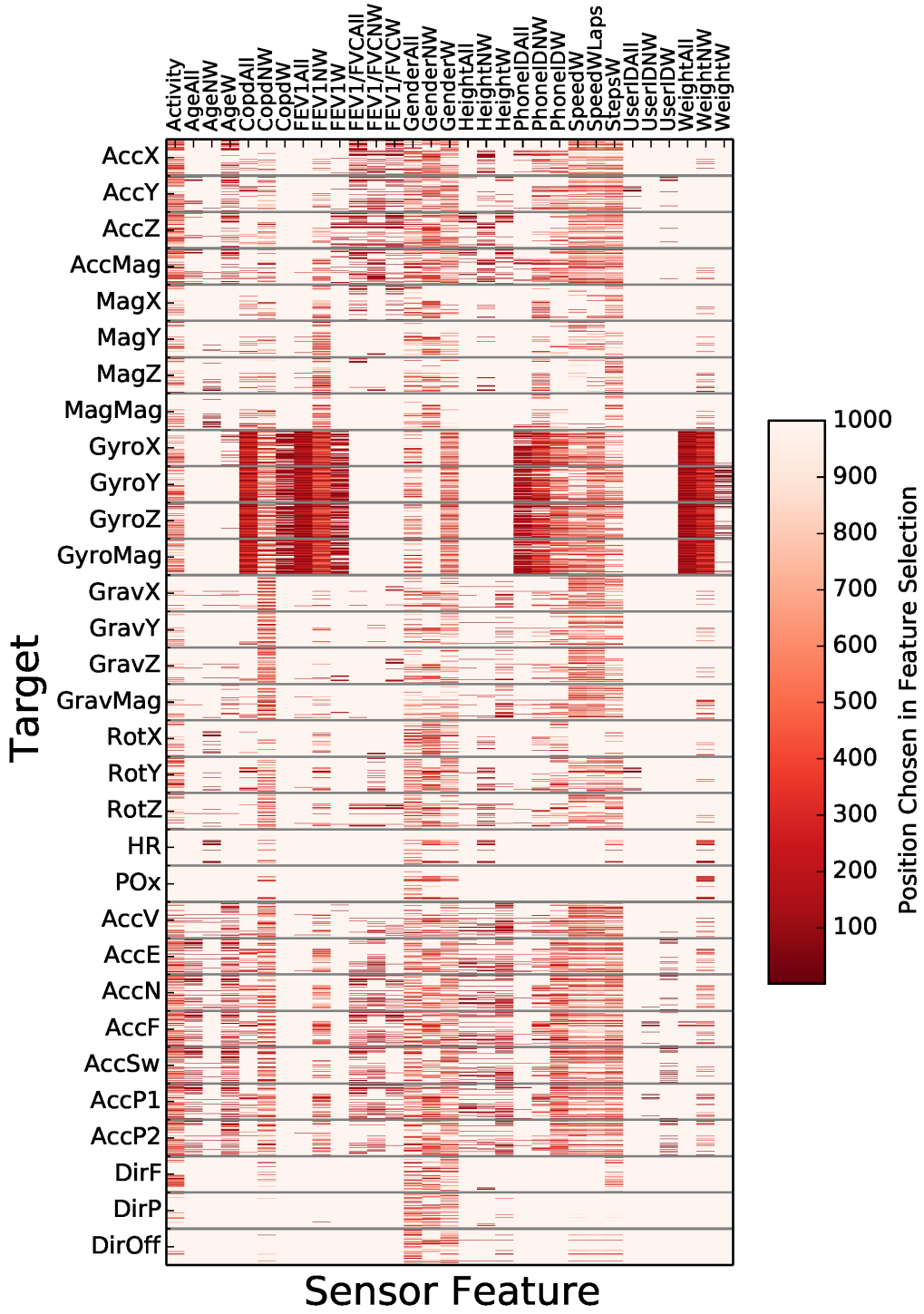

Figure 5.28: Top Scoring Private Features per Sensor Feature

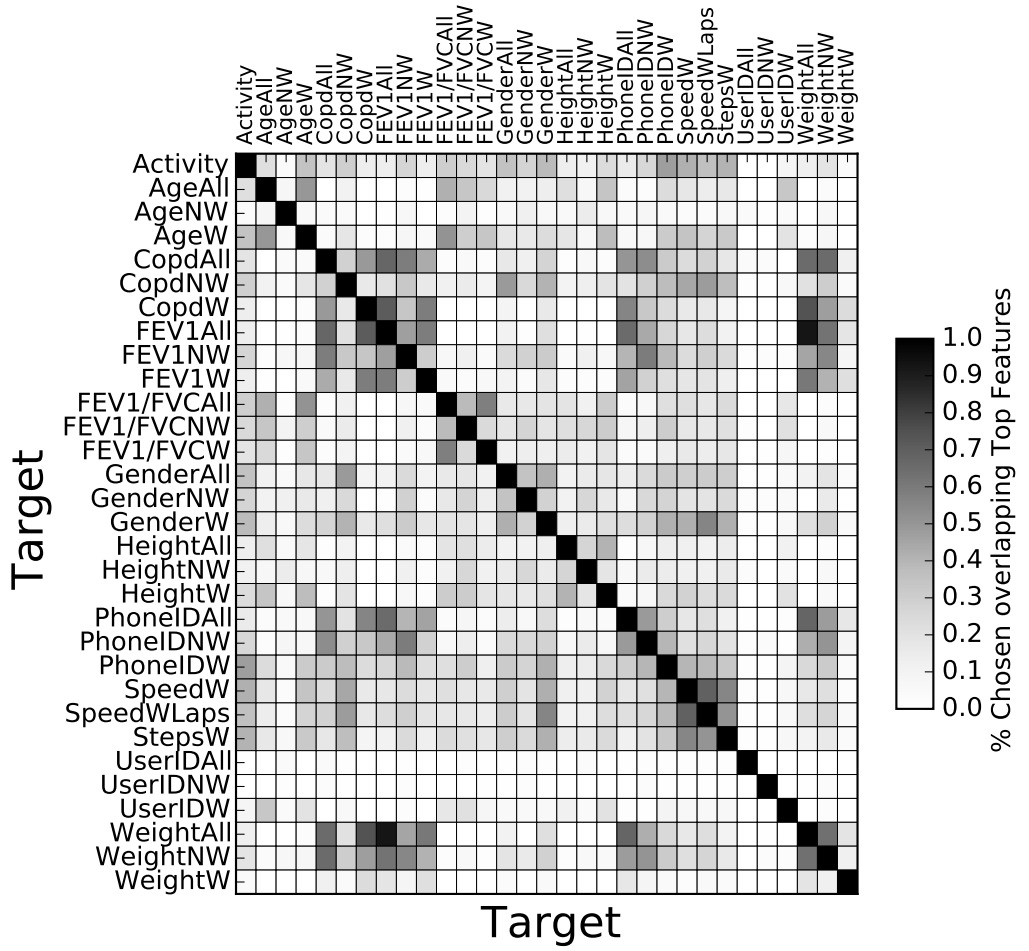

Figure 5.29: Intersection of Private Features among Targets

Comparing Private Features among Prediction Targets

Figure 5.29 gives a clearer picture of the intersection of private sensor features between predictions. We see that overall overlap varies per prediction target. Unsurprisingly, many prediction targets demonstrate strong correlation of private features between walking, non-walking and combined data sets. Optimistically, the overlap between user identification and other predictions is relatively small. This is primarily due to user identification having a smaller number of private features. Overall, we see a high rate of overlap between phone identification and health classification including FEV1% and COPD classification. Activity also overlaps with most predictive targets due to the high number of private features. Figure 5.30 demonstrates the relative number of private sensor features per predictive target. We see that many

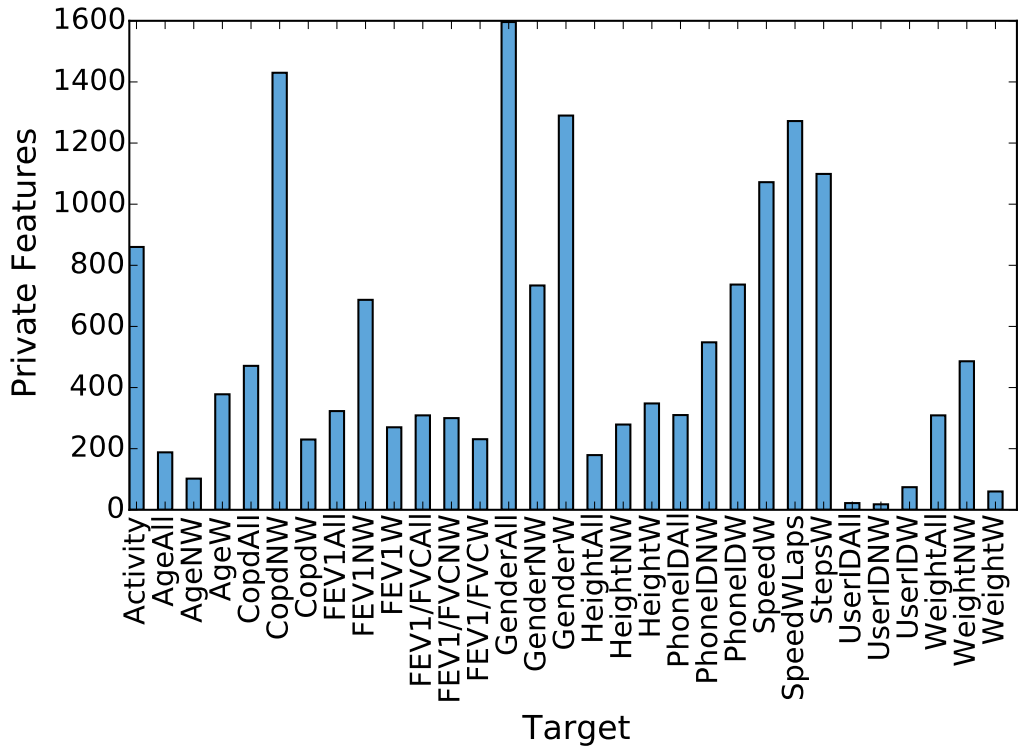

Figure 5.30: Number of Private Features per Target

predictive targets contain fewer than 500 private features removed to lower the predictive power below the 0.5 threshold. We notice that targets with an extremely high number of private features also attain low prediction accuracy. We surmise this is an indication of over fitting in the models which will be addressed in Chapter 6.

5.3 Conclusions

We have analyzed various feature selection routines in this chapter finding that filter and wrapper methods differ significantly in the top features identified. We investigated the effect of redundant features in top feature selection algorithms finding that subsets of features with high correlation can be identified and classified with similar levels of predictability. We then developed algorithms to identify the set of all private features which can be useful to conduct a prediction. While effective, our initial algorithms are computationally expensive requiring optimization. We therefore use NMI clustering

to reduce computation time evaluating the optimized and unoptimized algorithm for accuracy and runtime. We also evaluated the ability of traditional feature selection and feature evaluation statistics to identify private features. We find that using the original NMIS between features and target variables is the most accurate low computation method to identify the set of private features but still produces false positives and negatives against our baseline algorithm.

We determined the private features for all thirty predictive targets in our data set. We find significant overlap in both sensors and features requiring further analysis of the relative sensitivity between features for each target to determine the ability of obfuscating one prediction while leaving the other. We do identify thirteen features and three sensors not useful for predictions allowing us to reduce our total number of sensitive features by 26% to 1,708 sensor features. We determine the relative sensitivity of each sensor feature to give accurate predictions and design privacy obfuscation methods using this identified set of private features in the following chapters.

CHAPTER 6

PREDICTION MODELS

In this chapter, we consider various classification and regression models and evaluate their ability to predict sensitive information when trained with the top selected features presented in Chapter 5. Each classification model is trained using a given set of features X to predict a given target Y. For optimization, each model's hyperparameters are chosen using a grid search for the highest scoring model with reasonable boundaries depending on the model type. Once the ideal hyperparameters are found, the model is trained and cross validated using standard ten-fold cross validation. Once the model accuracy is determined, the chapter will finish by presenting novel methods to estimate the sensitivity of each feature in X to the target predictions Y. This analysis will be used in Chapter 7 to design privacy obfuscation techniques.

6.1 Classification Models

Classification models are designed to predict what class a group of input data belongs to or indicates. Each model is trained with a set of input parameters and target values. The model then determines cutoff points in the data to later classify unknown input as one of the previously trained classes. We now evaluate various classification models in their ability to predict classes including phone identification, user identification, gender, activity, and COPD status.

6.1.1 K-Nearest Neighbors

K-Nearest Neighbors or k-NN classification is perhaps the simplest algorithm in classification [91]. All training points are retained as a set of feature vectors with corresponding target values. When a new input feature vector

is presented, the algorithm searches for the k closest input feature vectors and predicts the target class using simple a majority vote of the most represented target in the set. The k parameter can be adjusted to optimize prediction accuracy to the underlying feature distribution.

6.1.2 Naive Bayes

Nave Bayes seeks to choose a classification target which reflects the prior observations used for training by defining an estimator $\widehat{y}$ which estimates the most likely y given an input feature vector x [92]. Formally, it looks for the best target y satisfying $P(y|x_1, x_2, ..., x_N)$ where the feature vector X contains N observations. We can represent this conditional probability using Bayes theorem (6.1). Of course, estimating the joint probability distribution given in (6.2) is not trivial. Thus, the "Naive" assumption of independence is assumed between all input features (6.3). This yields the simplification of the numerator with the conditional probability being replaced with the product of the independent conditional probability of each feature given the output target y. For our prediction of the best y, we can drop the denominator since the numerator is proportional to the joint probability (6.4). This yields the final estimator $\widehat{y}$ as the y that yields the maximum product of the probability of Y and the joint distributions of each feature value conditioned on the trained targets (6.5).

$P(y)$ is estimated using maximum a posteriori estimation which conveniently simplifies to the number of times the target y is observed divided by the total number of observations. The conditional probability can be modeled by a few distributions. Since the feature vectors in our training set are sampled from continuous distributions, we will model the conditional probability using the Gaussian distribution (6.6). This assumes the observations are sampled from a normally distributed continuous distribution. The paramaters μ_y and γ_y are calculated using maximum likelihood for each feature x_n for each target y.

$$P(y|x_1, x_2, ..., x_N) = \frac{P(y)P(x_1, x_2, ..., x_N|y)}{P(x_1, x_2..., x_N)} \tag{6.1}$$

$$P(x_n|y, x_1, ..., x_{n-1}, x_{n+1}, ..., x_N) = P(x_n|y) \tag{6.2}$$

$$P(y|x_1, x_2, ..., x_N) = \frac{P(y)\prod_{n=1}^{N} P(x_n|y)}{P(x_1, x_2..., x_N)} \tag{6.3}$$

$$P(y|x_1, x_2, ..., x_N) \propto P(y)\prod_{n=1}^{N} P(x_n|y) \tag{6.4}$$

$$\widehat{y} = \arg\max_{y} P(y)\prod_{n=1}^{N} P(x_n|y) \tag{6.5}$$

$$P(x_n|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}}\, e^{-\frac{(x-\mu_y)^2}{2\sigma_y^2}} \tag{6.6}$$

6.1.3 Quadratic Discriminant Analysis

Quadratic discriminant analysis uses the covariance between variables to estimate the particular class of a given set of inputs [93]. Once again, the variables are assumed to have a Gaussian distribution; however, no assumption is made of the independence of the features. We are again looking for the probability of a given target y given a set of x inputs. This can be represented by Bayes theorem (6.8). The probability of x, $P(x)$, is constant to all the comparative classifications and can be ignored. The probability of y, $P(y)$, is estimated using maximum likelihood estimation to be the relative frequency counts of the target y divided by the total observations (θ). The conditional probability distribution $P(x|y)$ is given by the multivariate Gaussian distribution (6.7) where k is the number of samples, μ is the mean of each feature x for the given target and $\sum$ is the covariance.

We can then define an estimator $\widehat{y}$ to predict the class. This corresponds to the maximum logarithmic a posteriori given by taking the logarithm of the multivariate Gaussian distribution multiplied by the prior. Once again, the $(2\pi)^{-\frac{k}{2}}$ term is a constant to all estimates and can be ignored. This gives the final terms of the estimator which includes a multiplication yielding an x^2 quadratic term (6.9). Classification is once again conducted by calculating (6.9) for every value of y and choosing the class with the greatest value (6.10).

$$f(x) = (2\pi)^{\frac{k}{2}}|\sum|^{-\frac{1}{2}}\, e^{-\frac{1}{2}(x-\mu)'\sum^{-1}(x-\mu)} \tag{6.7}$$

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} = \frac{f_y(x)\theta_y}{c} \tag{6.8}$$

$$\begin{aligned} \delta_y(x) &= ln(((2\pi)^{-\frac{k}{2}}|\sum|^{-\frac{1}{2}}\ e^{-\frac{1}{2}(x-\mu_y)'\sum^{-1}(x-\mu_y)}) * \theta_y) \\ &\approx -\tfrac{1}{2}ln|\sum|^{-\frac{1}{2}} - \tfrac{1}{2}(x-\mu_y)'\sum^{-1}(x-\mu_y) + ln\ \theta_y \end{aligned} \tag{6.9}$$

$$\widehat{y} = \arg\max_y\ \delta_y(x) \tag{6.10}$$

6.1.4 Decision Trees

Decision trees conduct classification by constructing a binary search tree with splits at each node made based on the input vector and the target of the classification in the leaf nodes [94, 95]. The algorithm iteratively builds the tree by recursively splitting the nodes until either the leaves only contains a single target or the tree reaches the maximum depth parameter. For each potential split, the algorithm calculates the fraction of targets which are contained in each potentially created leaf (6.11). The split is chosen with the minimum Gini impurity score (6.12). Intuitively, the Gini impurity score maximizes the number of unique targets contained in the split with an ideal score of zero having the split contain the complete subsets of each labeled target in the new leaf nodes.

$$f_i = \frac{1}{N}\sum_{k\in N} I(y_k = i) \tag{6.11}$$

$$H(f) = \sum_{n=1}^{N} f_n(1-f_n) = \sum_{n=1}^{N}(f_n - f_n^2) = \sum_{n=1}^{N} f_n - \sum_{n=1}^{N} f_n^2 = 1 - \sum_{n=1}^{N} f_n^2 \tag{6.12}$$

Classification is conducted by traversing the tree with the input vector X. The nodes are traversed according to the corresponding training rules and the predicted target $\widehat{y}$ is chosen at the reached leaf node. Decision trees work well at fitting the data exactly in noisy environments, but suffer severely from over-fitting in the presence of noise. Thus, two ensemble methods are often used in conjunction with decision trees including bagging and random forests [96, 97].

Bagging attempts to average the effect of over-fitting by training numerous decision trees on randomly sampled subsets of the training data with

replacement. The final classification is then conducted by taking the average probability for the answer from the generated trees.

Random forest decision trees also train multiple decision trees to reduce over-fitting. The training data is subsampled similarly to bagging, but the random forests train multiple decision trees per sample with each tree only training on a subset of the total number of input training vectors. Once again, the classification is conducted using the average probability of the answer over all the trees. This method also allows the importance of the individual features to be assessed by returning the score of the sub-trees which are trained on the individual features.

6.1.5 Support Vector Machines

Support vector machines are a class of machine learning algorithms which allow classification by remapping training data into a higher dimensional space and determining sets of hyperplanes that separate classes [98]. These hyperplanes called "support vectors" are calculated to maximize the separation between data points in the given space. Formally, the algorithm searches for the set of hyperparameters given by $\mathbf{w}$ which satisfies (6.13) subject to (6.14).

$$\min_{w,b,\gamma} \frac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{w} + C\sum_{n=1}^{N}\xi_n \tag{6.13}$$

$$y_n(\mathbf{w}^{\mathbf{T}}\phi(x_n) + b) \geq 1 - \xi_n,\ \xi \geq 1, n = 1, ..., N, y = \{-1, 1\} \tag{6.14}$$

Here $\mathbf{w}$ is the matrix of optimal support vectors. The mapping of the input vector of length N to higher dimensions is conducting by the ϕ operator. Outliers from the training data which may not be separable are handled with an error term ξ with C as an input parameter to the model controlling the penalty for incorrectly separated data points. This optimization problem is solved by introducing Lagrange multipliers and forming the dual (6.15) subject to (6.16). The decision function is then reduced to (6.17), where a positive sign indicates a predicted Y classification of 1 and a negative sign indicates a predicted Y classification of -1. While various choices for

kernels are possible, the two investigated in this dissertation are the linear kernel $K(x_n, x) = \langle x_n, x \rangle$ and the radial basis kernel $K(x_n, x) = e^{-\gamma|x_n - x|^2}$.

$$\min_{\alpha} \frac{1}{2} \sum_{m=1}^{N} \sum_{n=1}^{N} \alpha_m \alpha_n y_m y_n k(x_m, x_n) - \sum_{n=1}^{N} \alpha_n \tag{6.15}$$

$$\sum_{n=1}^{N} \alpha_n y_n = 0, 0 \leq \alpha_n \leq C \; \forall n = 1, ..., N \tag{6.16}$$

$$sgn(\sum_{n=1}^{N} y_n \alpha_n K(x_n, x) - \rho) \tag{6.17}$$

Obviously, the formulation given is a binary classifier. To conduct classification with more than two groups, a one-versus-one classification strategy is conducted with the final classification being the majority vote from the classifiers. Specifically, if there are k distinct targets, the classifier with train $\frac{k*(k-1)}{2}$ classifiers or one for each pair of unique targets. The prediction will then be the target that gets the most votes from among the classifiers.

6.2 Regression Models

Regression models predict the value of a continuous variable based on a given input vector. We will consider multiple types of regression models to predict the continuous targets in our data set including the age, weight, height, distance walked, FEV1% and FEV1/FVC.

6.2.1 Classical Regression

The simplest form of regressions fit a line directly correlating the inputs x and outputs y. These methods have the advantage of being extremely fast; however, input features that do not have linear relationships cannot be properly represented. Additionally, the linear methods can be extremely sensitive to noise in the input data. Thus, the simplicity often severely limits the utility of the linear regression methods. We evaluate a simple linear regression which chooses coefficients to minimize the ordinary least square problem as shown in (6.18).

$$\min_{w} ||Xw - y|| \tag{6.18}$$

Since we have noisy data, we also evaluate ridge regression which attempts to limit the effect of noise in creating unreasonably large coefficients by introducing a penalty as shown in (6.19) [99].

$$\min_{w} ||Xw - y|| + \alpha||w|| \tag{6.19}$$

Our final classical model is the Bayesian ridge regression which assumes the output y is represented by a multivariate Gaussian distribution centered around X and w (6.20) [100]. The values of the w coefficients are estimated using a prior of independent coefficients estimated from the data during training (6.21). The Bayesian ridge regression has the advantage of allowing for error during the training and estimation of the regression since the parameters are drawn from Gaussian distributions. This gives the model more versatility to fit the training data.

$$p(y|X, w, \alpha) = \mathcal{N}(y|X, w, \alpha) \tag{6.20}$$

$$p(w|\lambda) = \mathcal{N}(w|0, \lambda^{-1}\mathbf{I}) \tag{6.21}$$

6.2.2 k-Nearest Neighbors Regression

The k-nearest neighbors regression retains all points used during training [101]. To predict the output value, the k nearest training points are selected with the prediction being the average value among the k points. Similar to the k-NN classifier, the value of k is a user selectable hyperparameter.

6.2.3 Regression Trees

Regression trees are constructed in a similar manner to decision trees. Both methods construct a binary search tree with splits made at each node based on the target values in the training data for a given input vector [95]. Unlike the decision trees which split based on Gini impurity score, the regression trees choose the split to minimize the mean square error for each set of

training data represented in the new nodes (6.22). The final leaves hold the average values seen in the training data for their respective branches in the tree. Regression is conducted by traversing the tree choosing the split based on the given input vector X with the predicted output y being the averaged value in the leaf node. Regression trees can also be susceptible to over-fitting. As with decision trees, both bagging and random forest trees are ensemble methods which can help generalize the models.

$$\begin{gathered} H(f) = \frac{1}{N}\sum_{n\in N}(y_n - c_f)^2 \\ c_f = \frac{1}{N}\sum_{n\in N} y_n \end{gathered} \tag{6.22}$$

6.2.4 Support Vector Regression

Support vector regression (SVR) uses the same underlying theory used for support vector classification to predict value from a continuous distribution [98]. Instead of predicting the exact output value, the SVR predicts values within a margin of ε basically grouping output values within this margin to similar classes. Since the prediction predicts either above or below the mean, the margin of error term ξ is expanded to misclassification above the mean ξ and below the mean ξ^*. Once again, the method seeks a set of coefficients to minimize (6.23) under the constraint that the linear fit must be within the error term ε with C setting the penalty for misclassification. The dual optimization problem introduces the terms α and α^* (6.24). The prediction function is then given in (6.25) giving the estimate for the output value y.

$$\begin{gathered} \min_w \frac{1}{2}||w||^2 + C\sum_{n=1}^{N}(\xi_n + \xi_n^*) \\ y_n - wx_n - b \leq \varepsilon + \xi_n, \xi_n \geq 0 \\ wx_n + b - y_n \leq \varepsilon + \xi_n^*, \xi_n^* \geq 0 \end{gathered} \tag{6.23}$$

$$\begin{gathered} \min_{\alpha,\alpha^*} \frac{1}{2}(\alpha - \alpha^*)^T K(x_i, x)(\alpha - \alpha^*) + \varepsilon e^T(\alpha - \alpha^*) - y^T(\alpha - \alpha^*) \\ e^T(\alpha - \alpha^*) = 0, 0 \leq \alpha_n, \alpha_n^* \leq C, n = 1, ..., N \end{gathered} \tag{6.24}$$

$$y = \sum_{n=1}^{N} (\alpha_n - \alpha_n^*) K(x_n, x) + b \tag{6.25}$$

6.2.5 Gaussian Process Regression

Gaussian process regression predicts the output target by modeling the input features as sets of normally distributed random variables [102]. Gaussian process regression defines an estimator $\widehat{y}$ to estimate the output y after transforming the input features using a Gaussian function $h()$. It is assumed that the $\widehat{y}$ is also Gaussian with noise ϵ (6.26) giving a Gaussian $\sim \mathcal{N}(h(x_n), \epsilon_n)$ leaving the mean of the estimator $h(x_n)$. This assumes that $h()$ is also Gaussian such that $\sim \mathcal{N}(0, k)$ where k is the specified kernel function estimating the covariance between x values. $K(X, X_*)$ is defined to be a m by m kernel matrix such that $(K(X, X_*))_{ij} = k(x_i, x_{*j})$. Using these assumptions, we can define the Gaussian estimate of the distribution of input variables, the Gaussian estimate of the noise, and combine the estimates into an estimator of the output variables $\widehat{y}$ (6.27). Equation (6.28) follows directly from rules conditioning Gaussians. Therefore given an training set X and a set of corresponding kernel functions, we can obtain an estimate for the value of $\widehat{y}$.

$$\widehat{y_i} = h(x_n) + \epsilon_n, n = 1, ..., N \tag{6.26}$$

$$\begin{bmatrix} h \\ h_* \end{bmatrix} | X, X_* \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right)$$

$$\begin{bmatrix} \epsilon \\ \epsilon_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \sigma^2 I & 0 \\ 0^T & \sigma^2 I \end{bmatrix}\right)$$

$$\begin{bmatrix} \widehat{y} \\ \widehat{y}_* \end{bmatrix} | X, X_* = \begin{bmatrix} h \\ h_* \end{bmatrix} + \begin{bmatrix} \epsilon \\ \epsilon_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) + \sigma^2 I) & K(X,X_*) \\ K(X_*,X)) & K(X_*,X_*) + \sigma^2 I \end{bmatrix}\right) \tag{6.27}$$

$$\begin{aligned} &P(\widehat{y}_* | \widehat{y}, X, X_*) \sim \mathcal{N}(\mu_*, \sigma_*) \\ &\mu_* = K(X_*, X)(K(X,X) + \sigma^2 I)^{-1} \widehat{y} \\ &\sigma_* = K(X_*, X_*) + \sigma^2 I - \\ &\qquad K(X_*, X)(K(X,X) + \sigma^2 I)^{-1} K(X, X_*) \end{aligned} \tag{6.28}$$

6.3 Optimized Models and Prediction Accuracy

Previous work has used each of the presented models to conduct both classification and regression. We now evaluate each of our classification and regression targets using each model and feature selection method. While many previous works have been vague in presenting details about hyper-parameter selection and cross-validation methods, we seek to optimize each model type both in terms of hyperparameter selection and cross-fold validation. We will then present the results of the best feature selection and models which have been carefully optimized and trained.

6.3.1 Model Optimization

We test nine classification algorithms and nine regression algorithms for all targets. Each model is evaluated using a ten-fold cross validation which divides the data into ten equal pieces, trains on nine pieces, and evaluates on the held out piece. This process is repeated so every set of pieces is used once as evaluation and the results are averaged to attain the final $F1$-score (for classification) or mean absolute error (for regression). This process helps

to reduce overly optimistic estimates due to over fitting of the model.

Additionally, many models have various hyperparameters which need to be chosen. These parameters yield better models depending on the underlying data distributions of the training data. Since the distributions are largely unknown, it is common practice to train multiple models while sweeping over reasonable hyperparameters. The model with the best score is generally chosen and the same hyperparameters are used for further training. To reduce over fitting, we use a three-fold cross validation while training our models to select the ideal hyperparameters. For the support vector classification and support vector regression, we test both linear and radial basis kernels with C values of $0.001, 0.01, 0.1, 1, 10$ and 100 and gamma values with logarithmic steps from 10^{-12} to 0.1. We sweep over nearest neighbors for k-nearest-neighbor classification and regression from 10% to 90% of total samples used as nearest neighbors. Both bagged and random forest decision tree models are trained sweeping from 10 to 110 base trees. Linear regression is tested using an alpha value ranging from 10^{-5} to 10 in log space. Finally, the linear ridge regression is tested from 10^{-5} to 10 in log space for the alpha1, alpha2, lambda1 and lambda2 values.

The goal of the proposed analysis is to select the model most accurate for each prediction and to produce a cross-validated estimate which more accurately demonstrates the model's accuracy against over fitting. For each target, the nine classification or regression algorithms are first optimized using three-fold cross validation over all hyperparameters for the selected model. The optimal hyperparameters are then used to train a ten-fold cross-validated model. The cross-validated model with the highest classification score or lowest regression score is then selected as the optimal model for the prediction.

6.3.2 Choosing the Optimal Model

Figures 6.1 and 6.2 plot the percentage of times each model type is chosen when choosing the optimal model over all classification and regression optimizations. We see that the random forest classifier and support vector machine with radial basis kernels are most likely to have the highest accuracy for classification tasks. For regression, we find that the k-nearest neighbors

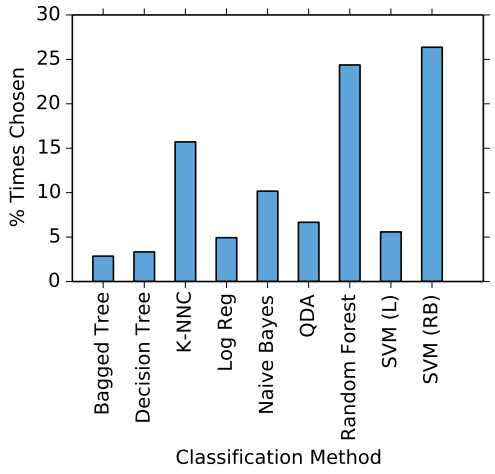

Figure 6.1: Top Classification Methods

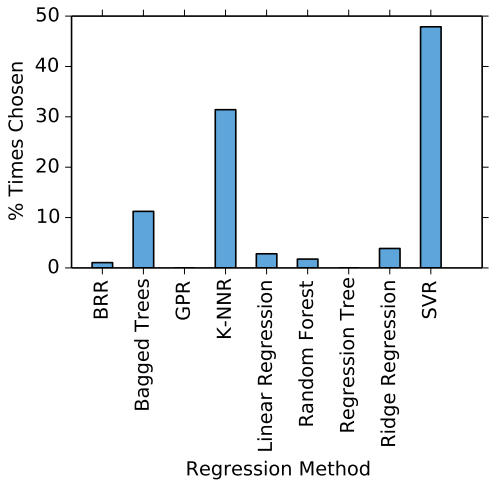

Figure 6.2: Top Regression Methods

and support vector regressions attain the lowest absolute mean error. Overall, the classification algorithms have more variation in the selection of top classifiers. The regression models are less varied with linear, Bayesian and tree based models having a much lower chance of being selected. It is particularly interesting that the decision trees, random forest tress, and bagged trees perform much poorer in regression tasks than classification tasks. This may be due to the added error of averaging between leaf nodes necessary during regression tasks.

While the SVM and random forest trees are generally the most accurate models for each task, the overall difference in accuracies between models for classification are actually fairly similar. This is unsurprising considering that many previous works have used each model to create accurate predictions. Figures 6.3 and 6.4 plot the min, average and maximum $F1$-scores of each optimized model for all feature selection methods. We see that the methods all predict activity with fairly high accuracies ($F1$-score greater than 0.9). The phone identification performs more poorly with the SVM and random forest giving the highest $F1$-scores of 0.8 and other models giving less accuracy ranging from 0.6-0.8. The accuracy per model type for speed is plotted in Figure 6.5. We notice that the Gaussian process regression performs extremely poor on outliers often returning absurd results. The figure is redrawn with a more reasonable axes in Figure 6.6. Once again, most models perform fairly well with the SVR yielding the lowest absolute error of 0.15 m/s. Thus,

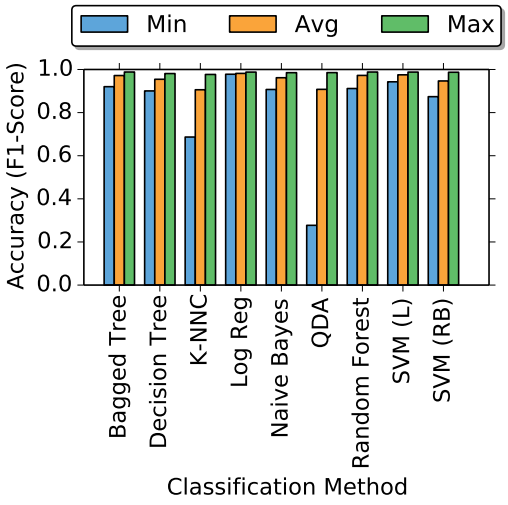

Figure 6.3: Activity Classification Model Accuracy

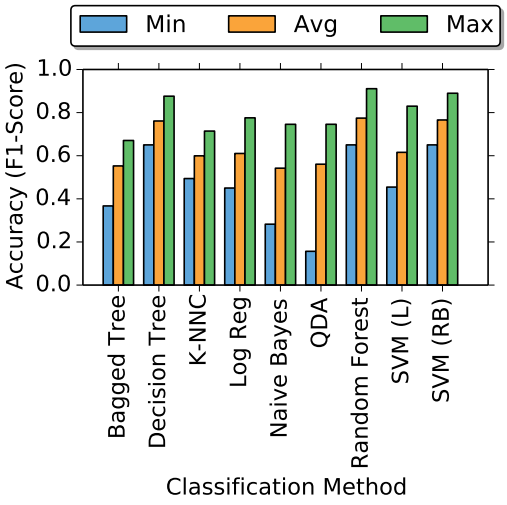

Figure 6.4: Phone Identification Model Accuracy

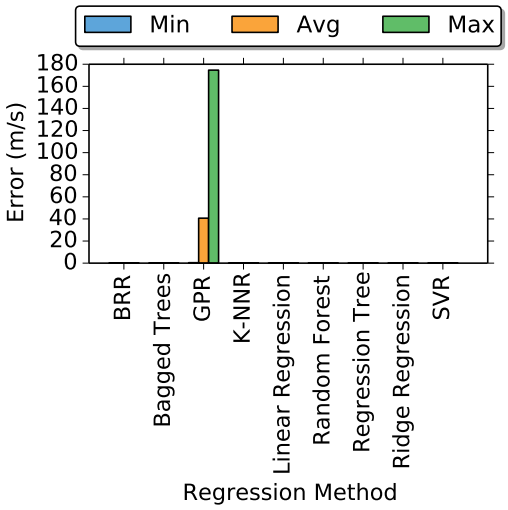

Figure 6.5: Speed Regression Model Accuracy

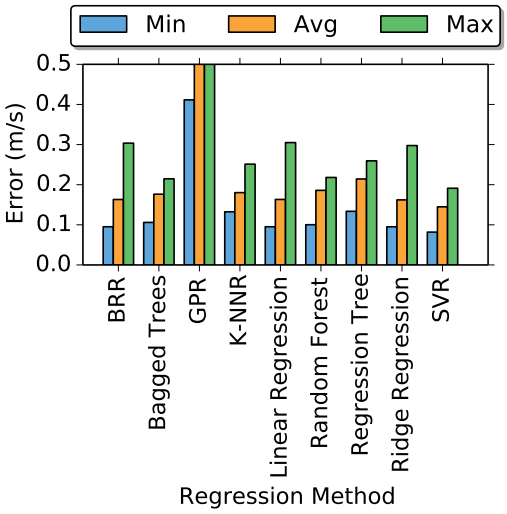

Figure 6.6: Magnified Speed Regression Model Accuracy

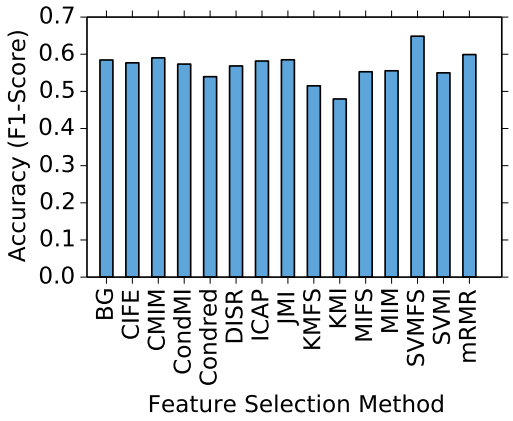

Figure 6.7: Average Best Classification Accuracy per Feature Selection Type

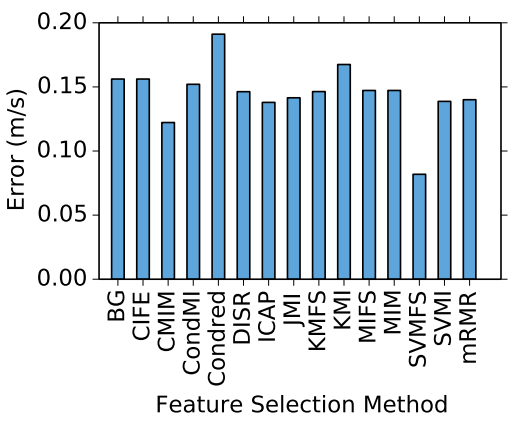

Figure 6.8: Best Prediction Accuracy for Speed Estimation

with exception of the GPR most models produce reasonable results when optimized; however, we find that support vector regression is most likely to be the best for regression tasks and support vector classification and random forest trees are most likely to be the best for classification tasks.

6.3.3 Comparing Feature Selection Accuracy

In Chapter 5, we found that feature selection routines varied widely in the sets of top features used for prediction. We now test the predictions with each of the fifteen feature selection techniques previously presented. Figure 6.7 lists the average best $F1$-score for each classification model over all classification targets. In general, most filter feature selection techniques appear to perform similarly. We see small performance gains in the JMI, CMIM and mRMR filter methods which affirms previous work which recommended the use of JMI as the default selection method [76]. We see that selecting the top features using k-means clustering performs worse than other methods which were tested. We also see that the SVM sequential forward search attains the highest accuracy overall. Since the scale of absolute average errors varies among regression methods, we plot a single regression, speed estimation, in Figure 6.8. In regression, we see more variation with the average performance in all cases. However, we still see that using the SVMFS wrapper method produces the lowest error in every model tested. Thus, we recommend run-

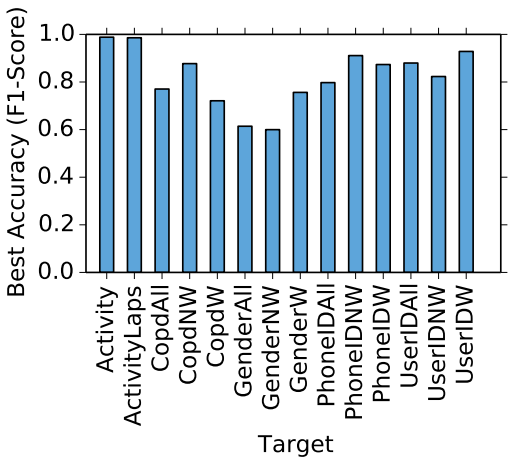

Figure 6.9: Best Prediction Accuracy for Classifications

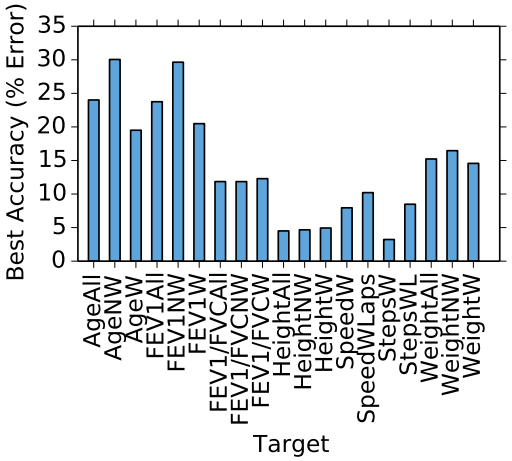

Figure 6.10: Best Prediction Accuracy for Regressions

ning the SVMFS for the highest accuracy. If computation power is limited, then we affirm that the JMI method overall produces the most accurate results for filter methods and recommend against using k-means clustering to select features.

6.3.4 Prediction Accuracies

After choosing the best feature selection technique and optimizing each machine learning model with 10-fold cross validation, we can now present an estimate of the prediction accuracies for each of our targets. Figure 6.9 demonstrates the $F1$-scores of each target. We find that predicting walking/non-walking activity is the most accurate with close to 99% accuracy over all samples using an optimized SVM with radial basis kernel trained using the features from the SVM sequential forward search. Phone identification is also quite accurate during periods of non-walking with scores above 0.91 for ten phones. COPD status classification attains an average $F1$-score of 0.87. User identification accuracy is quite low with walking identification $F1$-score of 0.31. Our sample size of 88 subjects is much larger than most previous work in subject identification. While previous work in user identification was conducted in carefully controlled laboratory settings, our walking includes many samples of free, unconstrained walking. However, we believe the decrease in accuracy is primarily caused by the ten-fold cross validation.

When testing so many subjects, the ten-fold cross validation can easily (and often) leaves out training for a subset of labeled targets. If we add the constraint that all ten-fold cross validation must have samples from all users, we attain $F1$-scores of $0.87, 0.93$, and 0.82 for user identification all, walking, and not walking respectively (as shown). We find gender attains low prediction scores. Gender prediction with an average of 0.61 barely attains better than a random binary classification.

Figure 6.10 presents the regression scores for the optimal regression models. We find that our methodology predicts speed with less than 0.08 meter per second error. We find that trained step regression models with roughly 1.5 step error over a ten second window perform more poorly than the anomaly detection techniques used in our own previous work. Our models predict the age of the subject within a ten-year period and the vital capacity within 11%. Unfortunately, the height had an average accuracy of three inches, the weight had an accuracy of thirty kilograms and the FEV1% scores roughly 20%. Since the ranges of target height had a standard deviation of 3.7 inches, weight 40 kilograms and FEV1% 14.7%, each model returns errors roughly equivalent to the standard deviation in the target's range making the predictions from the model statistically of little significance. Thus, we find our models can return predictions for speed, age, and FEV1/FVC with significant accuracy but are less accurate when predicting FEV1%, height, or weight. Steps are significant but less accurate than traditional step detection routines thus limiting the usefulness of the machine learning step models.

6.4 Estimating Sensitivity

Machine learning algorithms use sophisticated statistical models which often find underlying correlations in input vectors which are both non-trivial and non-obvious. While this makes the models useful for prediction, it also presents great difficulty in designing privacy techniques to protect the data. In order to protect the targets of the inference, we now present methods to estimate the amount of sensitivity S of an input vector given a machine learning model trained to predict a target T. Intuitively, knowing the average amount of change required in the input feature to affect the prediction target naturally leads to an intuition of the amount of obfuscation to maintain the

```
Result: List of Sensitivities
for training sample ts do
    for target t do
        for feature f do
            Target_distance[f] = distance from ts to decision boundary
        end
    end
    Training_sample_distance[f] = min(target_distance[f])
end
Sensitivity[f] = average(training_sample_distance[f]
```

Algorithm 4: Average Sensitivity for General Classifier

privacy of the target.

The following sections use the underlying theory of each model to design algorithms which estimate the sensitivity of the input feature to the prediction target. For classifiers, this is an estimation of the average distance of the input feature to the decision boundary. The derived algorithms calculate the distance for an individual feature calculated per training sample with the average distance over every training sample giving the estimate of sensitivity for each feature per target. In many cases, this can be represented by an equation giving a closed-form solution for the distance to the decision boundary. To estimate the sensitivity for the regression models, we estimate the ratio of change in input feature to change in output feature for each specific training point. The sensitivity will be defined as the change in output feature per unit change in input feature for each training point with the overall sensitivity being the average sensitivity per feature and target over all training points.

Numerous classification models internally use formulas or algorithms which allow direct calculation of the distance from a specified point to the decision boundary of the classifier. The general form of the algorithm used to get the average sensitivity when the distance is known is given in Algorithm 4. The sensitivity is calculated per feature as the average change in the feature over all training samples to change the classification. Of course, the amount of change required is the distance to the nearest decision boundary or the minimum across all possible decision boundaries. Thus the estimated sensitivity of a given feature is calculated to be the average minimum change required to cause the classification to differ from the true predicted value. The following sections will outline how the distances to the decision boundary

are extracted for each classifier.

6.4.1 Naive Bayes Classification

Recall that the estimator $\widehat{y}$ selects the classification with the maximum score for a given input point $x \forall n \in N$ as given in (6.29). Therefore, the decision boundary is the point at which the score for a given target y_1 is equal to y_2 (6.30). Substituting the Gaussian distance metric for the conditional probability and π_1 and π_2 for the maximum likelihood priors for y_1 and y_2, respectively, yields (6.31).

$$\widehat{y} = \max_y p(y) \prod_{n=1}^{N} p(x_n|y) \tag{6.29}$$

$$p(y_1) \prod_{n=1}^{N} p(x_n, y_1) = p(y_2) \prod_{n=1}^{N} p(x_n|y_2) \tag{6.30}$$

$$\pi_1 \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma_{1_n}^2}} \, e^{-\frac{(x_n - \mu_{1_n})^2}{2\sigma_{1_n}^2}} = \pi_2 \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma_{2_n}^2}} \, e^{-\frac{(x_n - \mu_{2_n})^2}{2\sigma_{2_n}^2}} \tag{6.31}$$

We are only interested in the change per feature f for which we are estimating the sensitivity. Fixing the other features and separating the terms we get (6.32). Taking the logarithm and re-arranging terms gives (6.33). This allows us to solve for the point x_f which is the distance from the training point to the decision plane for the feature f from the quadratic equation given in (6.34). Of course the final distance for feature n is the difference between the original training point x_n and the point on the decision plane x_f (6.35). The minimum distance between the training point x_n and each x_f for each available target is then taken as the minimum change required to cause a misclassification. This is then averaged over all training points for each feature to get the average sensitivity of each feature.

$$\pi_1 \frac{1}{\sqrt{2\pi\sigma_{1_f}^2}} e^{-\frac{(x_f-\mu_{1_f})^2}{2\sigma_{1_f}^2}} \prod_{n=1\neq f}^{N} e^{-\frac{(x_n-\mu_{1_n})^2}{2\sigma_{1_n}^2}} = \pi_2 \frac{1}{\sqrt{2\pi\sigma_{2_f}^2}} e^{-\frac{(x_n-\mu_{2_f})^2}{2\sigma_{2_f}^2}} \prod_{n=1\neq f}^{N} e^{-\frac{(x_n-\mu_{2_n})^2}{2\sigma_{2_n}^2}} \tag{6.32}$$

$$\frac{(x_f-\mu_{1_f})^2}{2\sigma_{1_f}^2} - \frac{(x_f-\mu_{2_f})^2}{2\sigma_{2_f}^2} = ln\left(\frac{\pi_1\sqrt{2\pi\sigma_{2_f}^2}}{\pi_2\sqrt{2\pi\sigma_{1_f}^2}}\right) - \sum_{n=1\neq f}^{N} \frac{(x_n-\mu_{1_n})^2}{2\sigma_{1_n}^2} + \sum_{n=1\neq f}^{N} \frac{(x_n-\mu_{2_n})^2}{2\sigma_{2_n}^2} \tag{6.33}$$

$$x_f^2(2\sigma_{2_f}^2 - 2\sigma_{1_f}^2) + x_f(4\sigma_{1_f}^2\mu_{2_f} - 4\sigma_{2_f}^2\mu_{1_f}) + 2\sigma_{2_f}^2\mu_{1_f}^2 - 2\sigma_{1_f}^2\mu_{2_f}^2 - ln\left(\frac{\pi_1\sqrt{2\pi\sigma_{2_f}^2}}{\pi_2\sqrt{2\pi\sigma_{1_f}^2}}\right) + \sum_{n=1\neq f}^{N} \frac{(x_n-\mu_{1_n})^2}{2\sigma_{1_n}^2} - \sum_{n=1\neq f}^{N} \frac{(x_n-\mu_{2_n})^2}{2\sigma_{2_n}^2} = 0 \tag{6.34}$$

$$d_n = abs\ (x_f - x_n) \tag{6.35}$$

6.4.2 Quadratic Discriminant Analysis

Quadratic discriminant analysis also allows us to derive a distance estimation from an arbitrary point to the decision boundary. Quadratic discriminant analysis chooses the classification that maximizes the discriminant function as given in (6.36) which is equal to the formula given in (6.37) after substituting the maximum likelihood prior estimate π_y, and taking the singular value decomposition of the covariance matrix to get the singular values S, the covariance inverse matrix R, and the input point $P = x - \mu$. In this formula, the summations over the matrices are written out with sums over all C target classes and F features. Once again, the point of interest is when

the value of the discriminant is equal for the given points or when P satisfies (6.38) for arbitrary targets 1 and 2.

$$\delta(y) = -\frac{1}{2} ln|\sum|^{-\frac{1}{2}} - \frac{1}{2}(x-\mu_y)'(\sum)^{-1}(x-\mu_y) + ln\ \theta_y \tag{6.36}$$

$$\delta(y) = -\frac{1}{2}\left(\sum_{c=1}^{C} ln(S) + \sum_{c=1}^{C}\left(\sum_{f=i}^{F} R_{fc} * P_f\right)^2\right) + ln\pi_y \tag{6.37}$$

$$\begin{gathered} -\sum_{c=1}^{C}\left(\sum_{f=i}^{F} R_{1_{fc}} * P_{1_f}\right)^2 + \sum_{c=1}^{C}\left(\sum_{f=i}^{F} R_{2_{fc}} * P_{2_f}\right)^2 = t \\ t = \sum_{c=1}^{C} ln(S_1) - \sum_{c=1}^{C} ln(S_2) + 2\ ln\frac{\pi_2}{\pi_1} \end{gathered} \tag{6.38}$$

Given an arbitrary point x, we are interested in the distance between the point and the decision plane. Letting $P_1 = x - \mu_1$ and $P_2 = x - \mu_2$ and adding a slack variable x_k representing the distance on the k feature axis of interest yields (6.39). Factoring out the x_k terms and defining convenience functions C_1 and C_2 yields (6.40). Rearranging terms, yields the quadratic equation given in (6.41). Once again, the distance is defined to be the difference between the given point x_n and the point on the decision plane x_k given in (6.42).

$$\begin{gathered} -\sum_{c=1}^{C}\left(\sum_{f=i\neq k}^{F}(R_{1_{fc}} * P_{1_f}) + (x_k + P_{1_k})R_{1_{kc}}\right)^2 + \\ \sum_{c=1}^{C}\left(\sum_{f=i\neq k}^{F}(R_{2_{fc}} * P_{2_f}) + (x_k + P_{1_k})R_{1_{kc}}\right)^2 = t \end{gathered} \tag{6.39}$$

$$\begin{gathered} -\sum_{c=1}^{C}(x_k * R_{1_{kc}} + C_{1_c})^2 + \sum_{c=1}^{C}(x_k * R_{2_{kc}} + C_{2_c})^2 = t \\ C_{1_c} = \sum_{f=i}^{F} R_{1_{fc}} * P_{1_f} \\ C_{2_c} = \sum_{f=i}^{F} R_{2_{fc}} * P_{2_f} \end{gathered} \tag{6.40}$$

$$x_k^2 \sum_{c=1}^{C} (R_{2_{fc}}^2 - R_{1_{fc}}^2) + x_k \sum_{c=1}^{C} (2R_{2_{fc}} C_{2_c} - 2R_{1_{fc}} C_{1_c}) + \sum_{c=1}^{C} (C_{2_c}^2 - C_{1_c}^2) - t = 0 \tag{6.41}$$

$$d_n = abs\ (x_k - x_n) \tag{6.42}$$

6.4.3 Decision Trees

```
Result: List of Windows
extractWindows(node, bounds);
if leaf node then
    if bounds != None then
        windows.add(bounds, node.target)
    end
else
    if node.feature == feature then
        windows = extractWindows(node.right_child, (bounds[0], node.threshold)
        windows = extractWindows(node.left_child, (node.threshold, bounds[1])
    else
        if sample[node.feature] < node.threshold then
            windows = extractWindows(node.left_child, bounds)
        else
            windows = extractWindows(nod.right_child, bounds)
        end
    end
end
return windows
```

Algorithm 5: Get Ranges of Feature Values That Predict Each Target

The average sensitivity using a decision tree model is once again calculated using Algorithm 4 to estimate the average variation per parameter. The distance or amount of change to change the classification is estimated using the recursive algorithm given in Algorithm 5. Given a root node in the tree, the algorithm searches the tree extracting the ranges of values of *feature* which predict each target assuming all other features in the given *sample* are kept constant. Thus, the algorithm will explore both sub-branches of the tree if the comparison feature of the node matches *feature*. The recursion down the left node sets the decision threshold of the node to the maximum bound of the window while the recursion down the right node sets the minimum

bound of the window. If the node does not make a decision based on the feature of interest, then the algorithm only explores the portion of the tree which would be selected based on the fixed features in *sample*. Of course, if the node is a leaf node, the current bounds and target set are added to *windows*. The final output is then the list of every set of bounds that will predict a specific target. This list is then parsed to determined which target in the set has the highest number of votes and merges adjacent windows with different target sets which would provide the same classification. Finally, the distance of the feature is set to the minimum distance between the original point of interest and the upper or lower bound of the window which contains the point.

The decision tree sensitivity algorithm can also be used to estimate the sensitivities of ensemble methods including bagging and random forest tress. In both cases, the average sensitivities of all sampled bagged decision trees and all subsampled random forest decision trees can be determined by taking the minimum, maximum or average values from the sub-trees in the classifier.

6.4.4 Support Vector Machines

SVMs are a binary classifier which uses a set of support vectors to calculate a given classification. We consider three popular implementations of the classifier including the SVC with linear and radial basis kernels which uses a set of one-versus-one classifiers and the linear SVC library which uses a set of one-versus-many classifiers. The one-versus-many method strategy used in the SVC routines creates a classifier which makes a binary decision between each target class. Thus, the decision point is when the prediction equals zero (6.43) with y, α and ρ being trained constants from the model and s_v being one of the V support vectors. We calculate the decision point with a linear kernel by plugging in the formula for the kernel (6.44) into (6.43). Once again, taking the constants from the trained model and setting all but the selected feature of interest as constants yields (6.45). Solving this equation gives the point of the decision plane for the feature of interest for the SVC model using a linear kernel (6.46). The distance is then calculated as the distance between the point of interest n and the decision point (6.47).

$$0 = \sum_{v=1}^{V} y_v \alpha_v k(\mathbf{s_v}, \mathbf{x}) + \rho \tag{6.43}$$

$$k(\mathbf{s_v}, \mathbf{x}) = \mathbf{s_v^T x} = \sum_{i=1}^{V} s_{vi} x_i \tag{6.44}$$

$$\sum_{v=1}^{V} y_i \alpha_i \left(\sum_{i=1 \neq f}^{F} s_{vi} x_i + s_{vf} x_f \right) + \rho = 0 \tag{6.45}$$

$$x_f = \frac{-\rho - \sum_{v=1}^{V} y_v \alpha_v \sum_{i=1 \neq f}^{F} s_{vi} x_i}{\sum_{v=1}^{V} y_v \alpha_v x_{vf}} \tag{6.46}$$

$$d_n = abs\ (x_f - x_n) \tag{6.47}$$

Once the distance from each individual one-versus-one classifier is determined, it is then possible to determine the minimum change of the feature value required to change the classification vote. This is done algorithmically as given in Algorithm 6. First, the current classification for the given point is determined. The total number of votes is then calculated for all one-versus-one classifiers. The votes for the current classification are stored in swingvotes. The list is sorted by ascending by distance. The algorithm then changes the swingvotes one at a time until the classification changes and returns the minimum distance necessary to change the vote.

The linear SVM method uses a one-versus-all approach to solve the direct SVC problem. This method while faster than the SVC method sacrifices some classification accuracy for speed and computational efficiency. Specifically, it chooses the class which yields the highest score given by the set of linear relationships in (6.48). Thus, the distance to each decision boundary is the value for x which makes the score for a given classification 1 equal to the score for a given classification 2 (6.49). Plugging in the trained values from the model for w and b and setting all but the feature of interest to constants yields an equation for the critical point of decision for a given feature (6.50). In the case of only two targets, the decision function is binary with only a single row of values for w and the critical point when the right side of (6.49) is zero. The decision point for this case is given in (6.51) and the distance

```
Result: Sensitivity
votes = array of votes per class
class = argmax(votes)
for each vote do
    if vote.winner == class then
        swingvotes.add(vote.distance, vote.winner, vote.loser)
    end
end
swingvotes.sort(ascending by distance) for each swingvote sv do
    votes[sv.winner] -= 1
    votes[sv.loser] += 1
    distance = sv.distance
    if argmax(votes != current) then
        break
    end
end
return distance
```

Algorithm 6: Calculate Sensitivity of One-versus-One SVC

between the point of interest and the decision point is again given in (6.47).

$$\max w^T \phi(x) + b \tag{6.48}$$

$$\sum_{i=1}^{F} w_{1i} x_i + b_1 = \sum_{i=1}^{F} w_{2i} x_i + b_2 \tag{6.49}$$

$$x_f = \frac{b_2 - b_1 + \sum_{i=1 \neq f}^{F} x_i (w_{2i} - w_{1i})}{w_{1f} - w_{2f}} \tag{6.50}$$

$$x_f = \frac{-b - \sum_{i=1 \neq f}^{F} x_i w_i}{w_f} \tag{6.51}$$

The final SVC model considered is the SVC with a radial basis function used as the kernel (6.52). Each decision function in this case yields (6.53). This gives a summation of exponentials and an unsolvable equation. Thus, the exact distance function cannot be calculated and we must instead use a generic estimation method as presented in Section 6.4.5.

$$k(\mathbf{s_v}, \mathbf{x}) = e^{-\gamma |\mathbf{x} - \mathbf{s_v}|^2} \tag{6.52}$$

$$0 = \sum_{v=1}^{V} y_v \alpha_v e^{-\gamma |\mathbf{x} - \mathbf{s_v}|^2} + \rho \tag{6.53}$$

6.4.5 General Classification Sensitivity Estimation

While some classification techniques allow direct calculation of the sensitivity of a feature by calculating the distance to the decision point, many other models are either impossible to solve such as the SVC with a radial basis kernel presented in Section 5.4.4 or lack a simple way to represent the decision boundary such as k-Nearest Neighbors (k-NN). Recall that k-NN classifiers work by taking the point to predict and finding the k nearest neighbors to the point. The output classification is then the majority vote of the k nearest neighbors. The decision plane between the points is therefore the point at which the majority vote changes.

To estimate the amount of sensitivity per feature for models which lack methods to calculate the distance to the decision point, we design Algorithm 7. The algorithm calculates the average change required in each input feature f required to change the classification over all training points p. It accomplishes this by first noting the total range difference between the maximum and minimum values for the feature. For each training point, it then recursively searches the positive and negative directions to find the distance to the decision boundary. The search is conducted by making successively smaller hops each time hopping half the previous distance until reaching a hop size less than a minstep, the stopping criteria passed to the algorithm. The distance for the training sample is taken to be the minimum distance to a change for both the positive and negative directions. The average required change is then taken over all changes calculated for all training points. This gives the average change in each feature which may change the output target prediction. While simple, the algorithm is unfortunately, slow compared to the closed-form solutions presented in the previous sections.

```
Result: List of Feature Sensitivities
for Each Input Feature f do
    range = max(f) - min(f);
    for Each Training Point p do
        step = range / 2.0;
        p' = p[f] + range;
        if model.predict(p') == model.predict(p) then
            s[f][p] = 0;
        else
            p' = p[f] + step;
            while step > minstep do
                step = step / 2.0;
                if model.predict(p') == model.predict(p) then
                    p' = p'[f] + step;
                else
                    p' = p'[f] - step;
                end
            end
            s[f][p] = abs(p[f] - p'[f]);
        end
        step = range / 2.0;
        p' = p[f] - range;
        if model.predict(p') != model.predict(p) then
            p' = p[f] - step;
            while step > minstep do
                step = step / 2.0;
                if model.predict(p') == model.predict(p) then
                    p' = p'[f] - step;
                else
                    p' = p'[f] + step;
                end
            end
            sneg = abs(p[f] - p'[f]);
            if sneg < s[f][p] or s[f][p] == 0 then
                S[f][p] = sneg;
            end
        end
    end
    S[f] = average(s[f][p] over p);
end
Return S;
```

Algorithm 7: Calculate Average Sensitivity for General Classifiers

6.4.6 Regression Sensitivity Estimation

Determining the sensitivity of a regression model to changes in an input feature is best accomplished by measuring the change in the output variable for various changes to the input. To determine this sensitivity, we present Algorithm 8. The function accepts an optional list of steps for manual calibration. If none is passed, the function will generate a list of steps to test between the minimum and maximum of each feature with a given minimum step size as the resolution. The algorithm will then iterate over all given sample points, over each input feature, and over each step each time calculating the absolute difference of the output divided by the step size. The algorithm then averages the value for all steps. This gives an estimate of the sensitivity per unit change in the input vector for each feature for each sample. The final sensitivity can then optionally be the minimum, maximum or average sensitivity over all given samples.

```
Result: List of Feature Sensitivities

if steps=None then
    for each feature f do
        steps[f] = range(min(f),max(f),minstep)
    end
end
for each sample s do
    basepredict = model.predict(s)
    for each feature f do
        numpredicts = 0
        for each step[f] st do
            test = s
            test[f] += st
            sensitivity[f] += abs(model.predict(test) - basepredict) / t
            Test[f] -= 2*st
            sensitivity[f] += abs(model.predict(test) - basepredict) / t
            numpredicts += 2
        end
        sensitivity[f] = sensitivity[f] / numpredicts
    end
    senssamples.append(sensitivity)
end
return sensamples
```

Algorithm 8: Calculate Average Sensitivity for General Regressors

6.5 Sensitivity Estimation and Results

We implement the methods presented in Section 5.4 using Python in our sensitivity estimation framework. The framework takes a trained model from the Scikit learning toolkit [103] and either a target point or set of training points and estimates the sensitivity of each individual feature. For an individual point, it returns the amount of change required to change the classification for classification models or the amount of change of output target per unit change of input feature for regression models. If a single point is passed, the framework returns the sensitivity of the single point. If the point is none, the framework will return the result of computing each training point with the answer either being the minimum, maximum or average distance depending on the input flags. Various methods also allow different averaging of internal distances (for example, the Naive Bayes can average the different combination of classifications). While configurable, the default methods are recommended. The framework automatically chooses the best method depending on the type of model passed to the calculate sensitivity function. Each method returns none, if there are no possible change in values for an input feature that yields a change in target prediction or the sensitivity is effectively infinite.

6.5.1 Types of Sensitivity

Sensitivity is useful in order to determine the amount of noise necessary to mitigate the threat of predicting privacy sensitive information. We present three types of sensitivity estimation. The most conservative sensitivity estimation is to introduce noise relative to the maximum difference between values of the input features. Intuitively, this makes it difficult to distinguish any two particular values from the input features but requires substantial noise since features may differ greatly between minimum and maximum values. Because traditional differential privacy uses this sensitivity estimate to provide a strong privacy guarantee, we call this noise estimate the differential sensitivity or diff sensitivity. We can also use our sensitivity estimation framework to estimate the sensitivity from the trained model but must carefully choose the trained models to use for the estimation. We use two types of models to evaluate the sensitivity. The first is to train SVM or SVR models

using each input feature individually. We call this the single sensitivity or S sensitivity estimate because it gives a sensitivity estimate of the individual feature to affect the output target prediction. Our final method is to train a SVM or SVR model using each of the top features using our fifteen feature selection routines and evaluate the sensitivity of each feature used in the model. We then average the sensitivity per feature to get the sensitivity of each feature giving an estimate of the amount of change per feature to affect the output of the prediction. We call this last sensitivity estimate the feature selection sensitivity estimate or FS sensitivity.

6.5.2 Sensitivity Estimate Testing

We conduct initial testing to demonstrate the different levels of sensitivity for our prediction targets including phone identification, user identification, FEV1/FVC, activity, and walking speed. We extract the set of all top features found using our fifteen features selection techniques. We train SVM or SVR models both for each individual feature and all features combined into one model. The maximum difference between feature values is used to estimate the differential sensitivity. The sensitivity estimation framework is used to extract the sensitivities for all individual feature sensitivities and for all combined models giving the single sensitivity and feature selection sensitivity estimates respectively.

Figure 6.11 lists the sensitivity values for each of the top features for the phone identification. We expect the maximum difference between feature values to be greater than any decision plane in a trained model. As expected, we see the highest sensitivity scores for the differential sensitivity estimates. Generally, we see the sensitivities found using the model trained with all features from feature selection have higher sensitivity than the estimates obtained from the individual models. Since the sensitivity measures the change required by an individual feature to change the output of the prediction model, a model with multiple features will require a single feature to be moved further to affect the decision of the model as a whole. Thus, the single sensitivity model which predicts the target solely dependent on a single feature yields the lowest estimate for sensitivity. Figure 6.12 presents the percentage of the differential sensitivity that is returned from both the

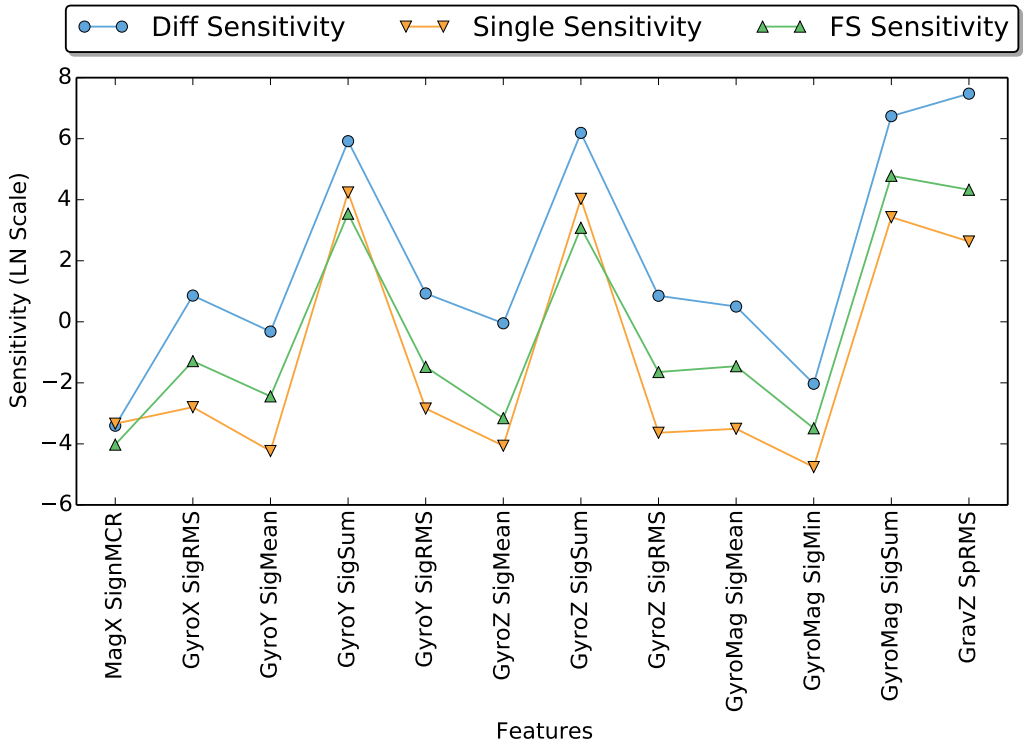

Figure 6.11: Phone Identification Sensitivity per Estimation Method (Log Scale)

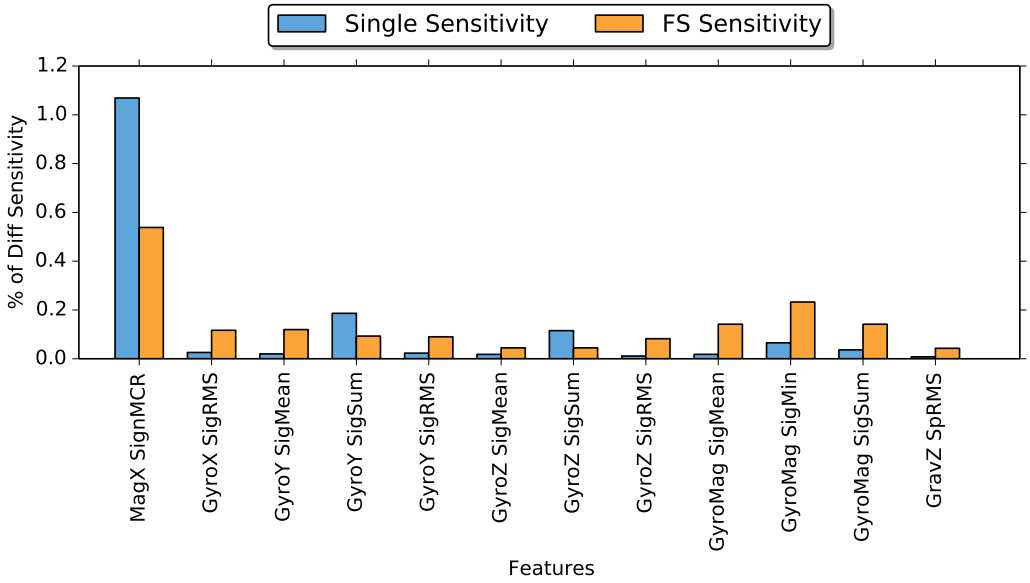

Figure 6.12: Phone Identification Sensitivity Fraction of Differential Sensitivity

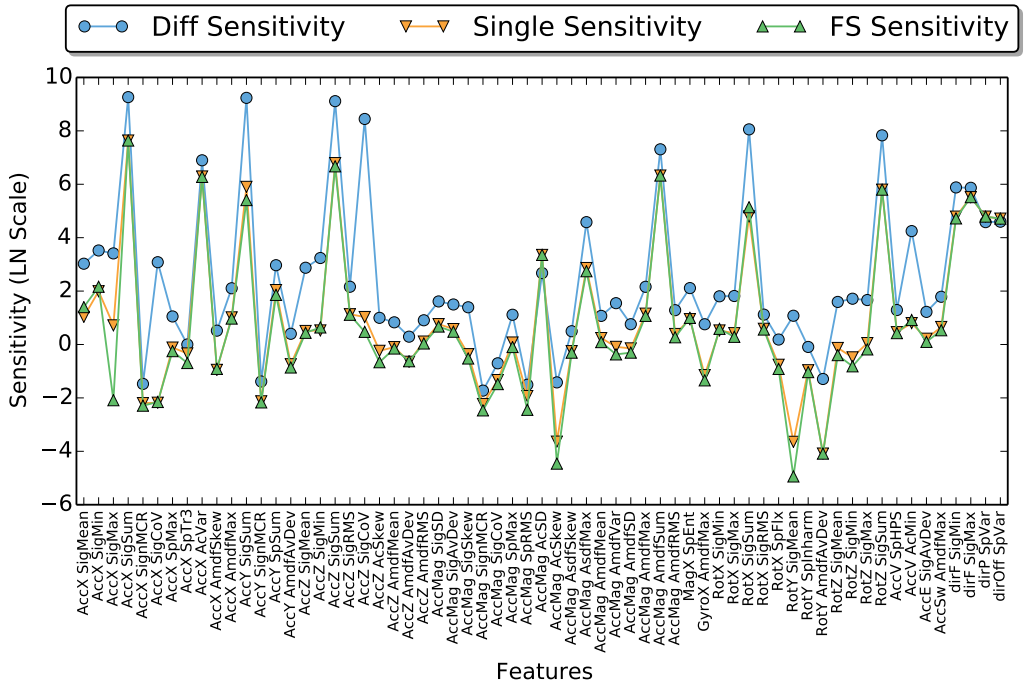

Figure 6.13: Speed Sensitivity per Estimation Method (Log Scale)

single and feature selection sensitivity estimates. We see that many features return sensitivities around 10% of the differential sensitivity estimates. Thus, using the sensitivity estimates requires noise roughly an order of magnitude less than the traditional differential private methods.

Figures 6.13 and 6.14 show the results of the three sensitivity estimates on the speed estimation. We note that the speed estimation has a larger set of top features returned from the feature selection methods due to less agreement of top features for this prediction. The sensitivity framework returns the amount of change in the input feature which produces a 1 m/s change in the output prediction. We see the same general ordering of features as in the phone classification example with both the single and feature selection sensitivity estimates returning an average of 20% the sensitivity of the differential private estimates. Interestingly, we do see outliers in the analysis, but find that most estimates with similar sensitivities between the differential and single/feature selection sensitivity estimates require absurdly high amounts of noise to change the output by one unit. Thus, features which are selected but do not greatly contribute to prediction still require high levels of noise. We surmise this is acceptable since these features can be filtered

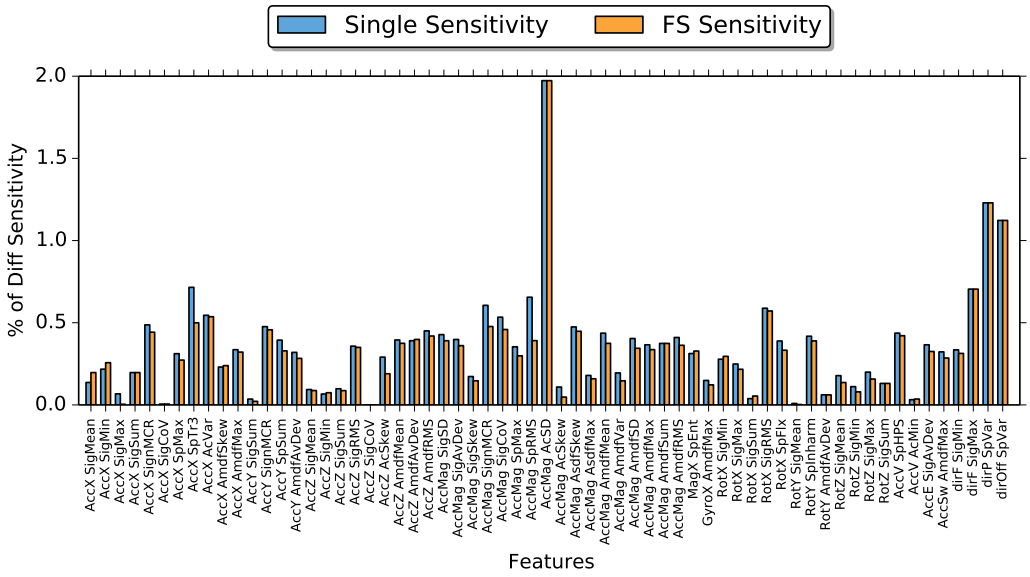

Figure 6.14: Speed Estimation Sensitivity Fraction of Differential Sensitivity

as unnecessary for obfuscation when protecting privacy because they have little effect on the regression estimates. The AccMag ACSD which actually returned a higher single sensitivity value than the differential private value presents an example of a feature with high noise requirements. We clearly see that high changes in this input feature caused little change to the output of the regression. Thus, the feature itself is a low priority for obfuscation since it contributes little to the speed prediction. Identifying features with high noise requirements that contribute little to the actual regression estimate is important to minimize the noise requirement during obfuscation.

6.6 Conclusions

In this chapter, we have explored the ability of machine learning models to predict a wide range of targets relating to health, fitness, demographic and devices using motion sensors from mobile phones. We used the feature selection techniques presented in the previous chapters to train optimal models for each target. Each model type was trained with optimized hyperparameters using three-fold cross validation. The prediction scores were evaluated using a ten-fold cross-validated training algorithm. We found that support

vector machines and random forest decision trees were most accurate for classification and support vector regression was most accurate for regression problems. However, we note that all the difference in maximum accuracy per classifier was much less than the difference between accuracies for feature selection. We surmise this similarity in accuracy helps drive the diversity of opinions in the literature in selecting the best machine learning tool. We find that the sequential forward selection wrapper method produces the best classification with the top score in nearly every case; however, it is fairly expensive. If computation power is limited, the JMI gives the best overall score of all tested filter methods. As a final note, we highlight that using a k-means clustering algorithm to choose features performed less accurately than all other feature selection techniques.

We developed a framework to estimate the sensitivity of a machine learning model's prediction to changes in an individual input feature. We present algorithms to estimate the distance to the decision boundary of a variety of classification and regression models. We implement our algorithms in a Python library to estimate the sensitivity of machine learning models trained using the Scikit learn library. We then compare the sensitivity of a machine learning model trained with the feature of interest and a machine learning model trained using the top features from feature selection against the traditional sensitivity estimate used in differential privacy. Overall, we see our framework giving sensitivity estimates which require substantially less noise for obfuscation than the differential privacy sensitivity. We will use these estimates to explore obfuscation techniques in Chapter 7.

CHAPTER 7

PRIVACY BY OBFUSCATING FEATURE STREAMS

In this chapter we design and evaluate methods to obfuscate feature streams to mitigate leaking information for specific inference targets. While we have seen the ability to predict multiple targets, we focus on realistic scenarios where privacy protection would be useful. We have seen that motion sensor data can be useful for phone identification and user identification. Phone identification has been proposed as a way to uniquely identify the device allowing targeted advertisements to track a specific device across sessions. User identification would allow a user to be tracked across devices and link all device activities to a specific individual. Both identifications are potentially sensitive, motivating the need to obfuscate the predictions from both health and fitness continuous monitoring. Thus, the first privacy scenario is to protect phone identification while predicting both health and fitness information. The second privacy scenario is to protect user identification while predicting both health and fitness information. There is clearly a distinction between basic fitness monitoring which includes activity recognition and steps taken and medical monitoring which includes precise speed measurement and medically accepted heuristics. Thus, our final scenario is to determine if we can hide health metrics while predicting fitness values. We consider activity using our walking or not walking binary classifier as the main fitness metric. We use the FEV1/FVC and medically accurate walking speed as our medical metrics. We note that speed could be considered a fitness metric; however, most fitness devices record speed with substantially less accuracy. Since medical studies have found strong correlation between speed and patient status, we surmise highly accurate speed requires similar protection as the standard medical measures. We now see which of our five targets can be obfuscated without significantly affecting the prediction of the other metrics.

We develop and analyze methods to reduce the ability of inferring various

targets from motion data based on our identification of private features and their respective sensitivity to noise. The most obvious way to stop prediction of sensitive features is to hide all private features for that prediction target. However, simply blocking features may cause significant collateral damage to the other prediction targets. By considering individual sensor features, we can directly apply ideas of differential privacy to determine the appropriate level of added noise required to block the ability to infer the private target without blocking the entire feature. Using the relative sensitivities determined from our sensitivity estimation framework, we can minimize the required noise thus minimizing the collateral damage. In the following sections, we will briefly review the idea of differential privacy and how we use it to determine the appropriate noise level for both classification and regression predictions. We will then look at the noise levels required based on the sensitivity estimates in Chapter 6. We will inject the appropriate amount of noise and verify the privacy produced against inference. We can then test the ability to infer other characteristics demonstrating how to selectively obfuscate target inferences while allowing the signal to be useful for other applications. We will then investigate the resistance of various machine learning methods to noise. We finish by assuming the attacker has prior knowledge of the noise levels used to obfuscate the features to determine if such knowledge would break a privacy system which adds noise. We conclude with a discussion of which of our privacy scenarios can be realized using the frameworks presented in this dissertation.

7.1 Obfuscation by Removing Private Features

The most straightforward method to protect privacy is to simply block all private features identified in the preceding chapters for each target of interest. However, since each prediction target has various overlapping features, this method may also significantly decreases the prediction accuracy of other targets. We evaluate the utility of blocking all private features by comparing models trained using the full set of features against models trained only using the non-private features. By comparing the prediction accuracy of the privacy sensitive target with and without the use of private features, we can determine the amount of privacy attained by blocking the features.

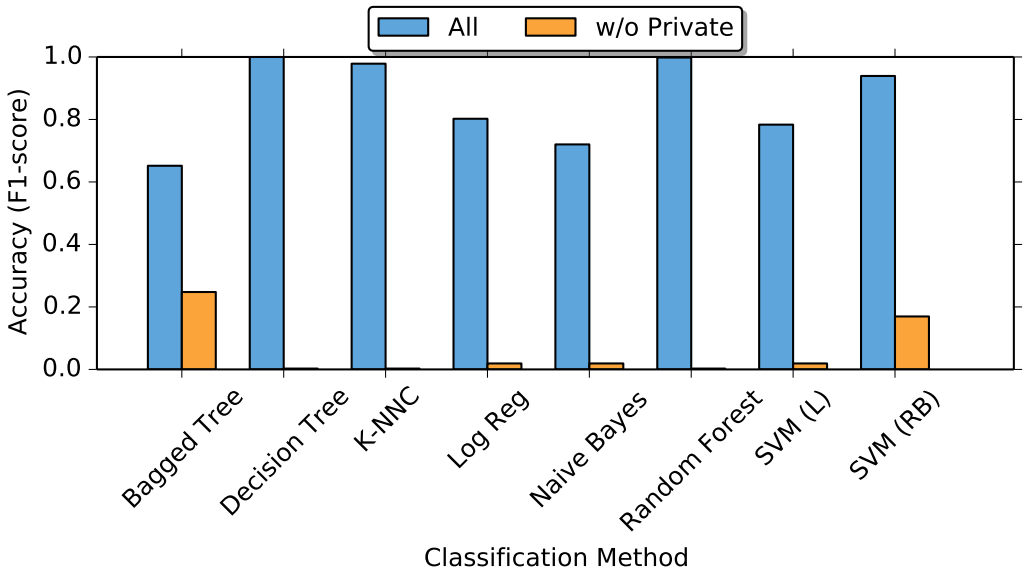

Figure 7.1: Phone Identification Accuracy with Phone Identification Private Features Removed over All Models

Models are then trained to predict the remaining four targets without the private features to determine the collateral damage of obfuscating the private target. This allows us to determine if blocking the private features destroys the ability to use the motion data for other predictions.

Figure 7.1 shows the prediction of phone identification for all classification models after removing all private features for phone identification. As expected, the ability to identify the phone is significantly diminished with only the bagged tree and SVM (RB) models giving any accurate predictions. Unsurprisingly, completely eliminating features classified as private significantly reduces the ability to predict our targets. We see two outcomes to remaining prediction targets when blocking all private features to obfuscate a prediction. For models with low feature overlap we see little performance difference between the prediction with and without the private features. Figure 7.2 demonstrates the effect of removing the private features identified for phone identification before doing the user identification. As expected, there is little drop in accuracy for the user identification since the removed features are not useful for predicting the user. It is important to note that the user identification has the least private features from our five target predictions. Furthermore, the phone identification is ideal since it favors features extracted from the gyroscope sensor in the phone while the other four targets

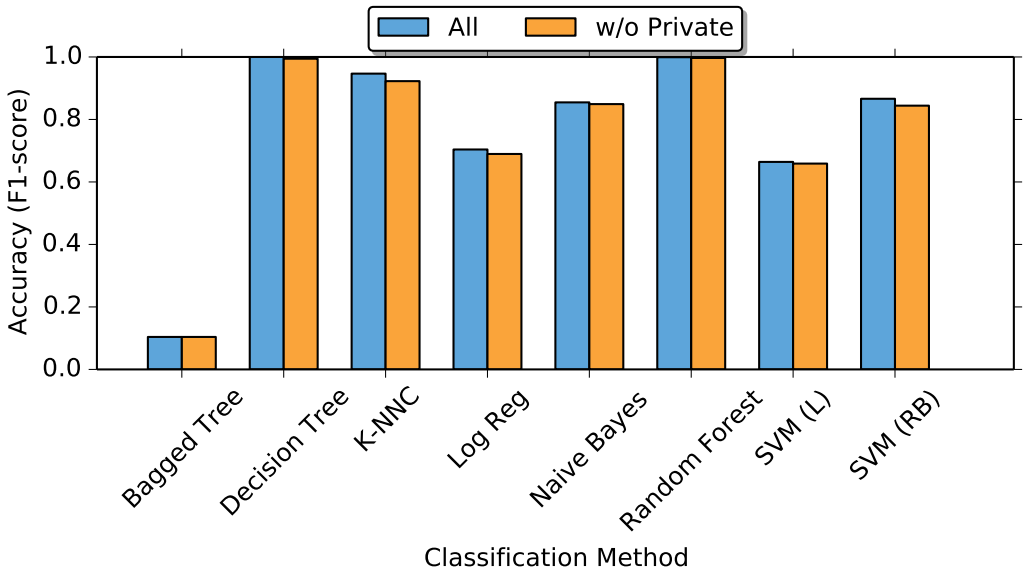

Figure 7.2: User Identification Accuracy with Phone Identification Private Features Removed over All Models

favor features extracted from the accelerometer.

Unfortunately, simply blocking the private features significantly decreases the performance of predictions with overlapping private features. Figure 7.3 compares the error rates for each model both with and without the private features useful to do phone identification. We see that removing the features useful for phone prediction significantly increases the error of the speed prediction. Thus, hiding the set of all private features does allow the phone identification to be protected without affecting targets with different private features such as user identification. However, for predictions which have overlapping private features including the speed, FEV1/FVC, and activity, removing the information causes the prediction to have unacceptable error rates. Prediction targets with overlapping features motivate the need to also investigate the sensitivity of the features and develop methods to introduce noise calibrated to the sensitivity of the features.

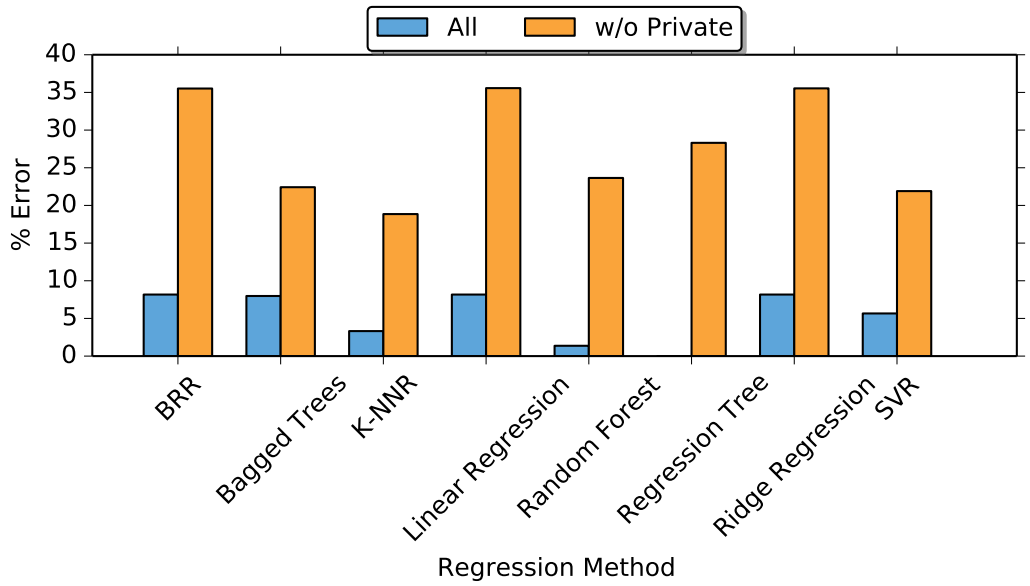

Figure 7.3: Speed Accuracy with Phone Identification Private Features Removed over All Models

7.2 Sensitivity between Prediction Models

The analysis in Chapter 5 demonstrated that many prediction targets share features which must be protected to obfuscate prediction. Thus, it is insufficient to simply drop the feature or block it in order to protect privacy without reducing the ability of other predictions. We expand on this analysis in this section using our sensitivity analysis framework presented in Section 7.1. We are specifically interested in determining if the features have similar sensitivity estimates per prediction. If a feature is highly sensitive when predicting target A but has much less sensitivity when predicting target B, then we can calibrate our noise level to hide A without significantly affecting the inference of B. In this case, we say that A and B can be obfuscated separately.

We refine our estimation of the obfuscation separability of targets A and B by taking the intersection of all identified private features using the analysis in Chapter 5. We then estimate the sensitivity using an individually trained SVM or SVR model with a radial basis kernel using the sensitivity framework presented in Chapter 6. This gives us the individual private sensitivity estimate for every overlapping private features for targets A and B. We then calculate the difference between sensitivity estimates as the percentage differ-

ence of the mean (7.1). Intuitively, features with similar sensitivity estimates or low scores will adversely affect both predictions when noise is introduced for obfuscation.

$$\text{Percentage Difference} = \left(\frac{abs(S_1 - S_2)}{mean(S_1, S_2)}\right) * 100\% \tag{7.1}$$

We look at the difference in overlapping sensitivity estimates for five features including the FEV1/FVC, phone identification, user identification, speed prediction and activity identification. For each target, we calculate the sensitivity difference for all overlapping features with the other four targets. Figure 7.4 shows the cumulative distribution function of the difference across all mutually intersecting private features. We see that the activity has many intersecting features with the other four predictions. We surmise this is due to activity being relatively easy to identify in most features extracted from the signal. Furthermore, these initial estimates imply obfuscating activity will most likely require a substantial amount of noise introduced into the signal which will affect intersecting predictions. Figure 7.5 plots the sensitivity intersection for the FEV1/FVC compared to the four other targets. We see that the FEV1/FVC shares similar sensitivity with a large number of features used to predict speed. The intersection with speed is interesting because it supports the suspected link between a subject's ability to walk and health status. Figures 7.6 and 7.7 illustrate the overlapping sensitivity between the phone identification and user identification to the four other targets respectively. We see few similar sensitivity estimates for both classifiers.

Figure 7.8 plots the CDF of the relative sensitivity of each feature selection method to the sensitivity of the activity classification given by (7.2). The diagram demonstrates that roughly 80% of user identification sensitivities and 70% of phone identification features sensitivities have lower sensitivity values and are more sensitive than activity recognition making them both more sensitive to noise than activity. Conversely, both the FEV1/FVCW and speed estimation feature sensitivities are generally less sensitive to noise than activity. We see similar patterns for FEV1/FVCW in Figure 7.9, phone identification in Figure 7.10 and user identification in Figure 7.11. Overall, we see that both phone and user identification have the lowest sensitivity values which should make them easier to obfuscate followed by activity with both walking speed and health identification having the highest sensitivity

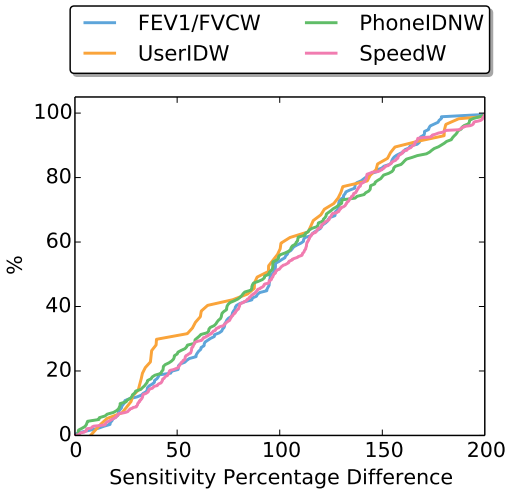

Figure 7.4: CDF of Activity Sensitivity Similarity

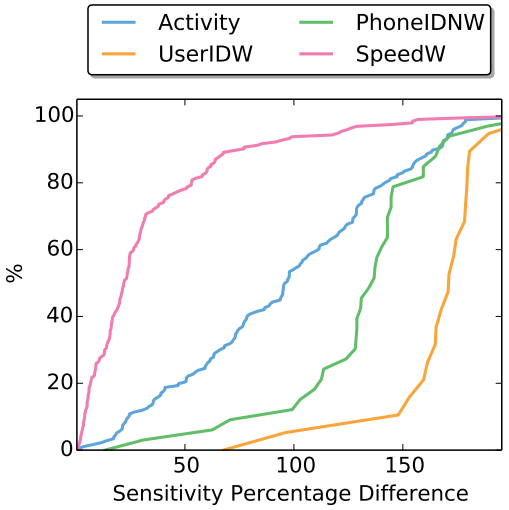

Figure 7.5: CDF of FEV1/FVC Sensitivity Similarity

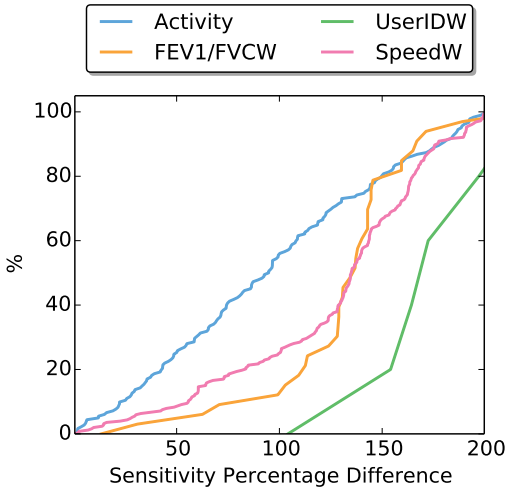

Figure 7.6: CDF of Phone Identification Sensitivity Similarity

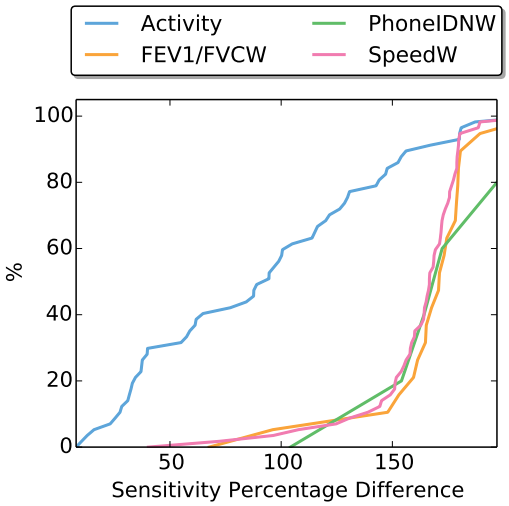

Figure 7.7: CDF of User Identification Similarity

values requiring more noise to obfuscate them from the signal.

$$\text{relative sensitivity} = \left(\frac{S_1}{S_T}\right) * 100\% \tag{7.2}$$

Initial sensitivity analysis indicates walking ability and health status appear to share similar sensitivity with overlapping training features. We see that activity recognition appears to have the greatest number of private features with enough sensitivity similarity to imply difficulty to obfuscate activity without negatively affecting other classifiers especially with user and

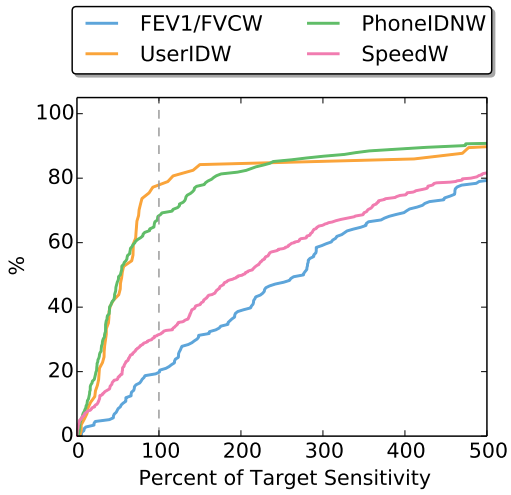

Figure 7.8: CDF of Sensitivity Relative to Activity Classification

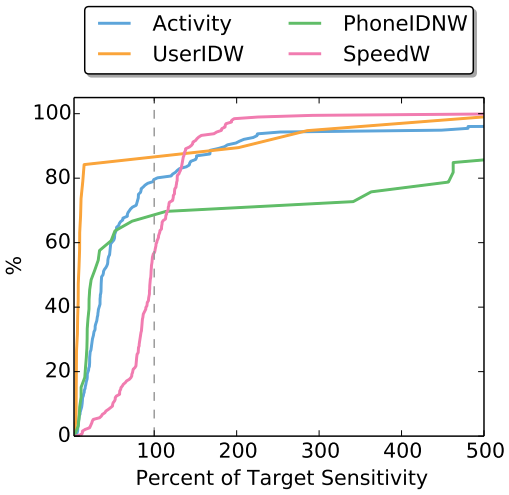

Figure 7.9: CDF of Sensitivity Relative to FEV1/FVC Regression

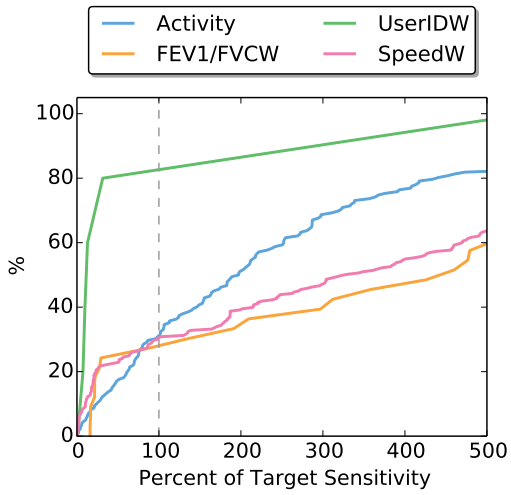

Figure 7.10: CDF of Sensitivity Relative to Phone Identification Classification

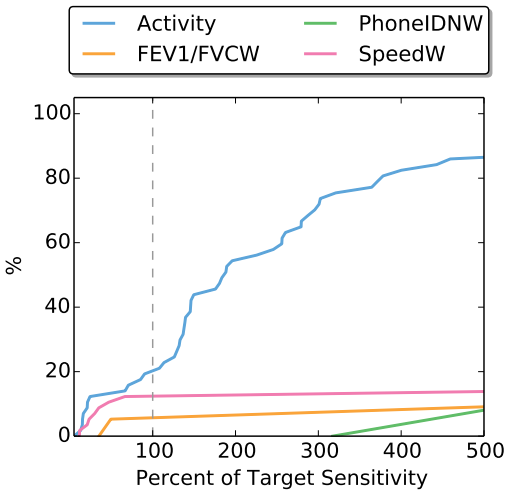

Figure 7.11: CDF of Sensitivity Relative to User Identification Classification

phone identification which are more sensitive to a majority of the overlapping features. Finally, both phone identification and user identification show lower sensitivity differences to other targets with lower sensitivity values implying they may be able to be obfuscated without affecting other predictions.

7.3 Differential Privacy Frameworks

$$P(A(D_1) \in S) \leq e^{\epsilon} P(A(D_2) \in S) \tag{7.3}$$

Differential privacy has been developed to produce strong guarantees of privacy to an individual when contributing data to a database [104]. The idea is to define a differential private query A which runs on a set of data entries D_k to produce a query result. Differential privacy guarantees that any two database entries that differ by one entry, D_1 and D_2, cannot allow an attacker to distinguish the database based on the query (7.3). In other words, the query will return a result which does not allow an attacker to determine the exact composition of the underlying database. This protects the identity of the individual records in the database. The classical way to design differentially private mechanisms is to introduce noise drawn from a Laplace definition proportional to the sensitivity of the release mechanism. The sensitivity is defined as the maximum difference between two sets of points that differ by one entry (7.4).

$$|f(x) - f(x')| \leq S(f) \tag{7.4}$$

While not directly applicable to sensor feature streams, the ideas of differential privacy have already been extended to other applications by Kifer and Machanavajjhala [105]. We will therefore use ideas from the differential framework to calibrate our obfuscation noise depending on the sensitivity of our classification techniques.

7.3.1 Classification Privacy

We first present the framework we will use to obfuscate feature steams to carefully decrease the ability to classify target characteristics. We assume we are measuring a feature $x \in X$. We define an obfuscation routine F that will release a point given x. Furthermore, we have a classification routine C that for some x will return a classification $c \in C$. We are interested in bounding the ability of an attacker to determine the correct classification c_k given the value x. We therefore define the classification privacy condition (7.5).

$$P(F(x)|C(x) = c_i) \leq e^{\epsilon} P(F(x)|C(x) = c_j) \tag{7.5}$$

We state that the probability of our obfuscation function $F(x)$ returning a value given the classification $C(x)$ being a specific class, is bounded by at most e^{ϵ}. This restriction is dependent on the sensitivity of change in classification to differences in x. Thus, the definition of the privacy metric is dependent on the trained model. The maximum sensitivity between any pair of classifications for a given pair of classifications is given in (7.6).

$$|(F(x)|C(x) = c_i) - (F(x)|C(x) = c_j)| \leq S(x) \forall x \in X, \forall i, j \in C \tag{7.6}$$

The preceding definition of sensitivity states that $F(x)$ must add enough noise that any two possible classification outcomes i, j must be equally likely. We also define a relaxed version of (7.6) where the sensitivity is defined as the distance required to guarantee at least two classifications are possible (7.7).

$$\min_{j \in C}(|(F(x)|C(x) = c_i) - (F(x)|C(x) = c_j)| \leq S(x) \forall x \in X, \forall i \in C) \tag{7.7}$$

Intuitively, this relaxation may be reasonable to trade off sensitivity versus privacy. For example, in a database with many distinct users, the motion data may be able to classify each individual user. However, there will most likely be subsets in the user population. Perhaps males form a distinct subset and females form a distinct subset. Equation 7.6 requires that every user has enough obfuscation to be indistinguishable from every other user which would require a significant amount of noise. Equation 7.7 allows us to relax this constraint to forcing the ϵ differential privacy between any two users. Thus, the user is not uniquely identifiable but is still afforded some privacy. Since we are dealing with a one-dimensional data set, we maintain the differential privacy guarantee by adding noise sampled from a Laplace distribution (7.8).

$$F(x) = x + n(x), n(x) = laplace\left(\frac{S(x)}{\epsilon}\right) \tag{7.8}$$

7.3.2 Regression Privacy

Protecting features from regression analysis is slightly more complicated than protecting classification. In the most trivial sense, any obfuscation will affect the output from the regression. However, since most regression algorithms use some form of continuous estimation function, it is expected that a similar value in the input feature will yield a similar value in the output feature. We build and expand on previous work applying the ideas from differential privacy to obfuscate location privacy in order to protect against values inferred from regression [106, 107, 108]. Our goal is to define a privacy mechanism that does not allow an adversary to know the true value of a regression better than a certain privacy range. For example, if the regression algorithm predicts a user's weight, we may define our privacy distance d as 50. If the regression outputs the weight in kilograms, this means an attacker should not be able to be confident that the feature data yields information which the regression would predict the weight to within 50 kilograms. Thus, the policy driven value d is selected as the distance to which the attacker is uncertain about the predicted value. We note that the distance metric has a unit in this case kg. To make the input of privacy dimensionless, we assume the unit of ϵ is the inverse of whatever unit of measurement under consideration. Thus, in the formulations when we multiply $\epsilon * d$, we get a dimensionless constant.

We begin by defining the distance of uncertainty around the input feature. We estimated this in Chapter 6 as the amount of change in the feature required to affect a corresponding change in the output from the regression. This means that the relationship between the distance of the feature d_f and the distance of the output from the regression d_r is related by $d_r = \frac{\Delta R}{\Delta F} d_f$. We want to intuitively provide privacy for a change in d_r but the units of x are relative to d_f. Since we define the sensitivity of the regression to be change in regression per change in feature, $\Delta F = 1$ and we want our distance to be $d_f = d_r * (\Delta R)^{-1}$. Note that ΔR is a constant of the regression model being evaluated. We will use this in each of the formulations below.

We first place a limit on the amount of information that releasing a feature value f can leak about the probability of yielding a specific value. Ideally, we would like the observation f to not give any further information about the relative probability of the possibility of two values. In other words, we

would like $\frac{P(x|f)}{P(x)} \leq e^{\epsilon}$. However, this is far too strong of a condition since knowing the range of values will give the attacker some information. We therefore define a function $B_d(x)$ to be the range of all possible values less than d_f away from the point x. We are only interested in the ability of an attacker to distinguish values within this range. Thus, we define our privacy distance $d_f = d_r * (\Delta R)^{-1}$ and say that any two points within this distance leak a bounded amount of information (7.9). The proof that this condition is satisfied by a differential private mechanism is defined in [107].

$$\frac{P(x|f, B_d(x))}{P(x|B_d(x))} \leq e^{\epsilon \Delta R d_r^{-1}} \;\; \forall \; d_r > 0 \;\; x \in X \tag{7.9}$$

Classical differential privacy has strong guarantees about prior information. Ideally, the ability of an attacker to infer one point over another given a released observation would be bounded $\frac{P(x|f)}{P(x'|f)} \leq e^{\epsilon}$. Unfortunately, the output from a regression means that the attacker may know certain values are far more likely than others before any information is released. In other words $\frac{P(x)}{P(x')}$ may be large. In this case, we want to guarantee that releasing f does not significantly change the ability of the attacker to discern x versus x' (7.10). We note that this equation is trivially obtained from (7.11) using Bayes rule.

$$\frac{P(x|f))}{P(x'|f)} \leq e^{\epsilon \Delta R d_r^{-1}} \frac{P(x)}{P(x')} \;\; \forall d_r > 0 \;\; \forall x, x' : |x - x'| \leq d_r (\Delta R)^{-1} \tag{7.10}$$

Finally, we define the bounds on choosing the output value f to that the value f does not distinguish between two point x and x' given that the distance between the two points are within the range d_f as shown in (7.11). Notice that this equation closely resembles the classical differential privacy form by some simple algebraic manipulation (7.12).

$$\frac{P(f|x)}{P(f|x')} \leq e^{\epsilon \Delta R d_r^{-1}} \;\; \forall d_r > 0 \;\; \forall x, x' : |x - x'| \leq d_r (\Delta R)^{-1} \tag{7.11}$$

$$P(f|x) \leq e^{\epsilon \Delta R d_r^{-1}} P(f|x') \;\; \forall d_r > 0 \;\; \forall x, x' : |x - x'| \leq d_r (\Delta R)^{-1} \tag{7.12}$$

We add noise to the feature set in order to satisfy the above equations by

again sampling from the Laplace distribution. We sample with a standard deviation of $\frac{d_f}{\epsilon} = \frac{d_r}{\epsilon \Delta R}$ in order to add the appropriate noise. Thus, we only need to choose the range of interest d_r and the relative privacy guarantee ϵ. $DeltaR$ is a constant estimated from the regression models from Chapter 6. We then sample noise according to (7.13). We will design experiments and evaluate this method in the following sections.

$$f = x + n(x), n(x) = laplace\left(\frac{d_r}{\epsilon \Delta R}\right) \tag{7.13}$$

7.4 Protecting Private Information

We now test our obfuscation techniques to hide five prediction targets (three classification and two regression models) including phone identification, user identification, activity recognition, FEV1/FVC, and walking speed. We implement a function to introduce noise drawn from a Laplace distribution into the feature vector. We obfuscate the ability to conduct each prediction by introducing noise into all privacy sensitive features produced by the sequential forward selection with clustering and similarity scoring metrics introduced in Chapter 5. The noise level is estimated using the three sensitivity estimation techniques produced in Chapter 6 including the traditional differential privacy estimate, the single trained model estimate and the feature selection model estimate.

We first evaluate and compare the various sensitivity estimation techniques. We then look at the overall ability to obfuscate prediction targets without affecting the other predictions by adding noise using the single model method with various privacy ϵ values. After adding noise, the ability of all five models is then re-evaluated to determine the accuracy reduction to predict the private feature as well as the effect on the prediction accuracy of the other models. In our final test, we investigate the ability of alternate prediction models to resist our obfuscation techniques by comparing the reduction in accuracy for each model type after adding noise. We also train the models using a noisy signal to simulate the ability of an attacker to improve models by taking the privacy obfuscation into consideration.

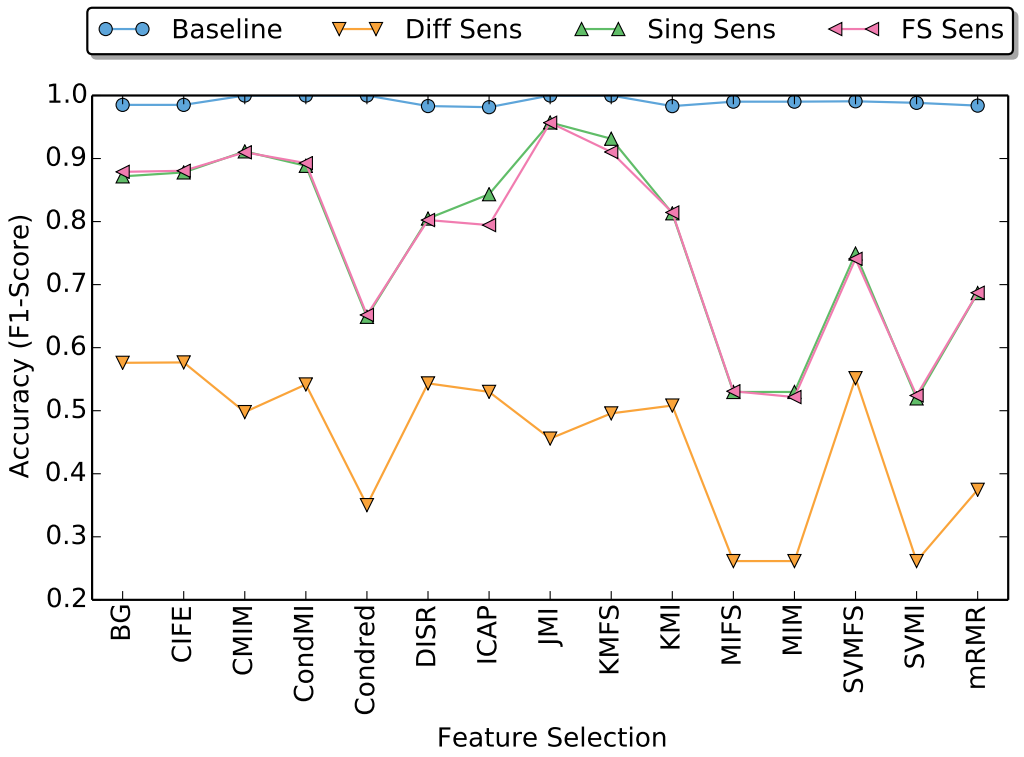

Figure 7.12: Phone Identification Accuracy with Phone Identity Obfuscated with Various Sensitivity Estimates

7.4.1 Estimating Sensitivity

We first use our testing to compare the three sensitivity estimation techniques presented in Chapter 6. Figure 7.12 plots the best case classification accuracy across all trained models of phone identification after obfuscating the private features according to the sensitivity estimates needed to hide the identity of the phone with an ϵ value of 1.0. As expected, the pure differential privacy which introduces noise proportional to the maximum difference between any two feature values produces the lowest classification score when trying to identify the phone after obfuscation. We see that both sensitivity estimates derived from the single sensitivity model and feature selection sensitivity models yield similar levels of privacy. Overall, we see some variation in the effectiveness of the noise to provide privacy guarantees across feature selection techniques. In the phone classification, the various noise estimates produce prediction scores which are proportionally similar. All three methods do significantly decrease the $F1$-scores and decrease the ability to predict phone identification.

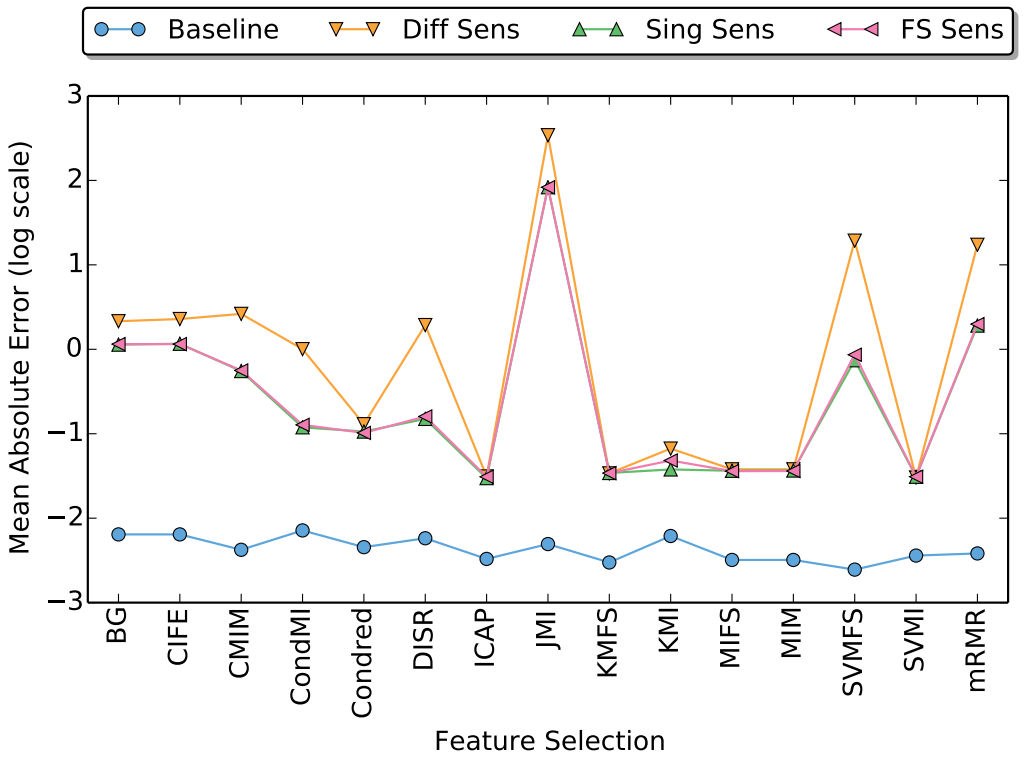

Figure 7.13: Speed Identification Accuracy with Speed Obfuscated with Various Sensitivity Estimates

Figure 7.13 demonstrates the best case prediction of speed over all regression models when the speed is obfuscated with a distance of 1 and ϵ of 1. Once again, the differential sensitivity estimate provides the greatest amount of obfuscation providing the most obfuscation. The single model and feature selection sensitivity estimation methods again provide similar prediction scores after obfuscation. While the baseline error is on the order of 0.1 m/s, the minimum error after obfuscation is on the order of 1 m/s. Thus, the method appears to satisfy (7.12) by making the points difficult to predict within a 1 m/s interval. This implies that in this test, the error in the regression estimate is relatively low. We finally note that the maximum amount of obfuscation is introduced in the two most accurate feature selection routines, the JMI and SVMFS.

We note that both graphs demonstrate that the methods work across various feature selection even though the feature obfuscation is estimated from the private feature identification algorithm. This algorithm does not contain the same set of features as most of the filter method selection algorithms.

This can explain some of the variation in prediction results across feature selection algorithms. We find that obfuscation is most effective when the set of features in the feature selection are contained in the privately identified features. This should not be problematic as long as the features not identified as private cannot predict above the threshold used to identify private features. We see this working especially in the speed identification example where the ICAP, KMFS, and KMI identify fewer than three private features. Since the remaining features do have predictive usefulness, albeit less than is required to be classified as a sensitive feature, the prediction score is higher than the feature selection routines which choose all obfuscated features. But, since the features are less predictive the speed prediction is still within the bounds of the privacy mechanism. We finally point out that both the single sensitivity models and feature selection models produce similar levels of degradation in the privacy predictions; however, the single sensitivity model requires significantly less noise per feature as was demonstrated in Chapter 6.

We have seen that the three sensitivity methods can obfuscate a prediction target with various levels of effective privacy. In the classification case, we saw the differential sensitivity giving better privacy, but in the speed regression, many feature selection methods actually had similar privacy even though the differential method introduces an order of magnitude more noise. Figure 7.14 shows the obfuscation from all three sensitivity methods when used on the optimally trained phone classification model only. We see that each method reduces the accuracy of phone classification to roughly 50% using an epsilon value of 2. It is noteworthy that once again, privacy is maintained using the single sensitivity estimate which requires significantly less noise. Figure 7.15 demonstrates that using too much noise can lead to interference when classifying other targets. This figure shows the classification accuracy of user identification when the same noise is introduced into the features that was used to generate Figure 7.14. We see that the accuracy of user identification is least affected when using the single sensitivity estimate with the classifier almost being useless when using the diff sensitivity estimate. Thus, we use the single sensitivity estimate for the remainder of this chapter.

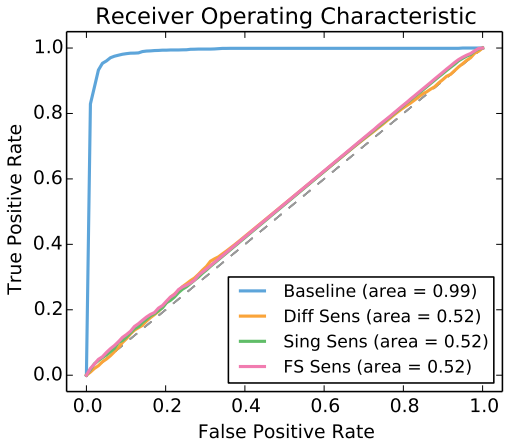

Figure 7.14: Phone Identification Accuracy with Phone Identity Obfuscated with Various Sensitivity Estimates

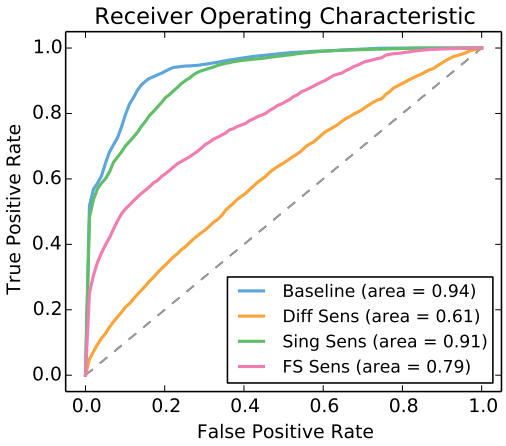

Figure 7.15: User Identification Accuracy with Phone Identity Obfuscated with Various Sensitivity Estimates

7.4.2 Effectiveness of Obfuscation and Collateral Damage

We now investigate the effect on each model's predictive power when we obfuscate a prediction target. To keep the experiments tractable, we only consider the optimized top model returned from the model optimization from Chapter 6. This means that each model uses the top scoring cross-validated model with the top-rated feature selection method. The base score is the score returned from the fully trained model. We then sweep over each optimized model predicting the output target with various values of epsilon including $2.0, 1.0, 0.5$ and 0.1. We repeat the testing for each obfuscation target giving five targets and five assessments totaling twenty-five tests.

Figure 7.16 compares the effects on activity prediction using various sensitivity levels to obfuscate against activity classification. We see that without noise, activity classification is nearly perfect. With a large ϵ value, the accuracy of activity classification is hardly affected. With a low value of ϵ, we see activity classification drop to nearly random guessing for the binary classifier. Figures 7.17 and 7.18 show the effects on predicting both phone and user classification with activity obfuscated. We see that phone identification is not diminished when hiding activity but user identification is significantly decreased. Tables 7.1 and 7.2 list the error rates of the regressions with activity obfuscated. We see a significant increase in error for both regressions with activity obfuscated even at low levels. Thus, we see activity as

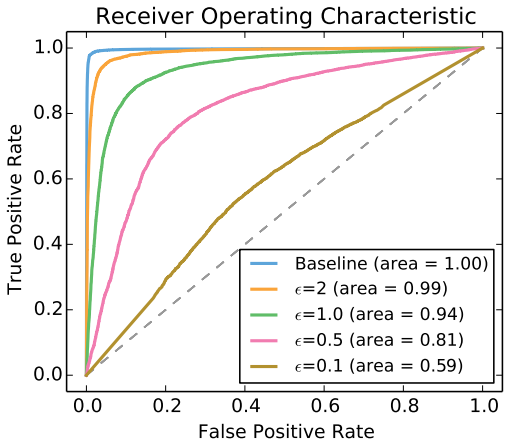

Figure 7.16: Activity Accuracy with Activity Obfuscated

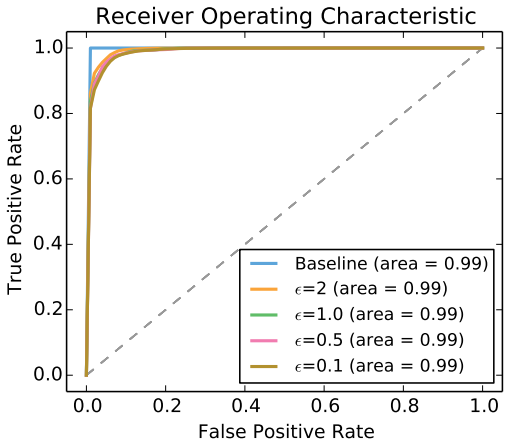

Figure 7.17: Phone Identification Accuracy with Activity Obfuscated

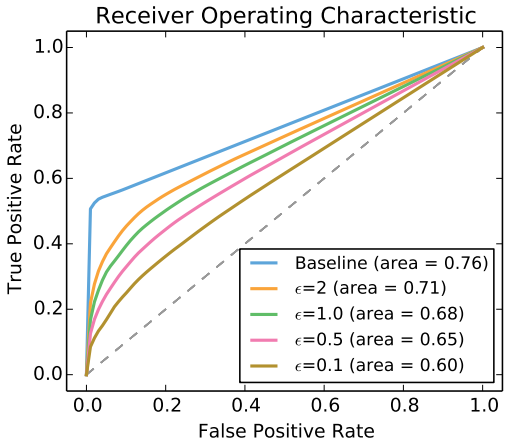

Figure 7.18: User Identification with Activity Obfuscated

difficult to hide without significantly decreasing all classifiers except phone identification which seems to have a disjoint set of predictive features.

Table 7.2 presents the relative speed regression accuracy with various levels of obfuscation for speed. We see the effect of varying the distance parameter most clearly in the speed regression. A change in ϵ is equivalent to the inverse of the distance. Thus, epsilon values of $2, 1, 0.5$, and 0.1 correspond to distance values of $0.5, 1, 2$, and 10. We see this directly in the mean noise estimates with the speed obfuscation returning a mean error of $0.53, 0.98, 1.90$, and 7.20 respectively. Thus, we see that obfuscation working well for our privacy levels. Obfuscating speed significantly hinders our other predictions. Figures 7.19, 7.20 and 7.21 demonstrate that obfuscating speed significantly decrease classification accuracy for activity, phone identification

Table 7.1: FEV1/FVCW Accuracy with Various Obfuscations

Obfuscated Target	Privacy ϵ Values .1	.5	1	2
FEV1/FVCW	33.30	15.69	13.58	12.55
Activity	46.74	18.44	14.85	13.00
PhoneIDNW	20.93	12.78	11.76	11.21
SpeedW	37.84	16.68	13.94	12.47
UserIDW	10.58	10.54	10.52	10.50

Table 7.2: SpeedW Accuracy with Various Obfuscations

Obfuscation Target	Privacy ϵ Values .1	.5	1	2
SpeedW	7.20	1.90	0.98	0.53
Activity	7.85	2.20	1.20	0.63
FEV1/FVCW	5.99	1.45	0.76	0.43
PhoneIDNW	1.60	0.63	0.34	0.20
UserIDW	0.52	0.24	0.15	0.11

and user identification respectively. We also see significant error introduced to FEV1/FVCW prediction as seen in Table 7.1. Thus, obfuscating speed also obfuscates the other targets of prediction. We also see that speed and FEV1/FVCW are linked with both being difficult to obfuscate without negatively affecting the remaining predictions.

We find that both phone and user identification require lower levels of noise to hide the signal than activity, speed and health estimates. Figure 7.22 shows the relative phone prediction accuracy with the phone identification obfuscated. We see that both activity and user identification predictions are hardly affected with the introduction of noise to obfuscate the phone identification as shown in Figures 7.23 and 7.24. We see that both speed estimates and FEV1/FVC are adversely affected with the introduction of obfuscation; however, the introduced error is roughly half of the introduced error from both activity and speed obfuscation. Figure 7.25 presents the user prediction accuracy after obfuscating the user identification. For user identification, we see a decrease in phone identification accuracy in Figure 7.26 but almost no effect on activity recognition as shown in Figure 7.27. We also see the least effect on both regression estimations with Table 7.1 showing almost no effect on FEV1/FVC prediction and the smallest effect of

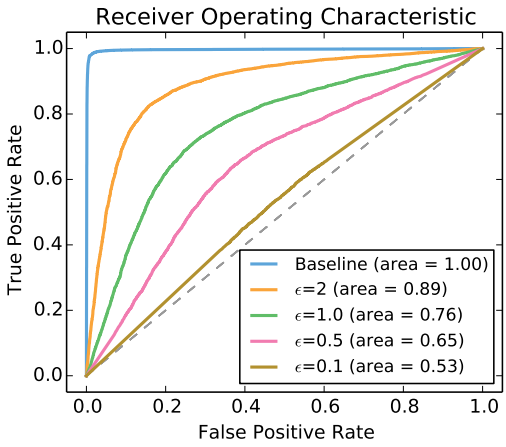

Figure 7.19: Activity Accuracy with Speed Obfuscated

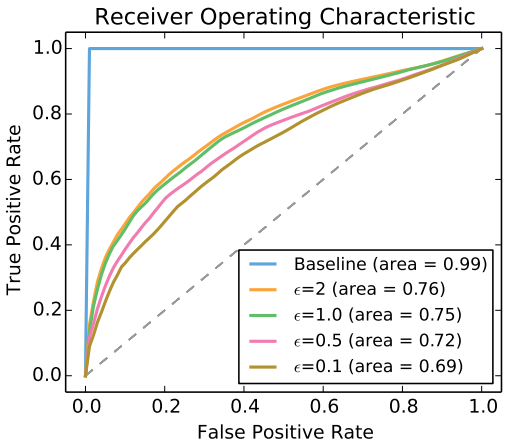

Figure 7.20: Phone Identification with Speed Obfuscated

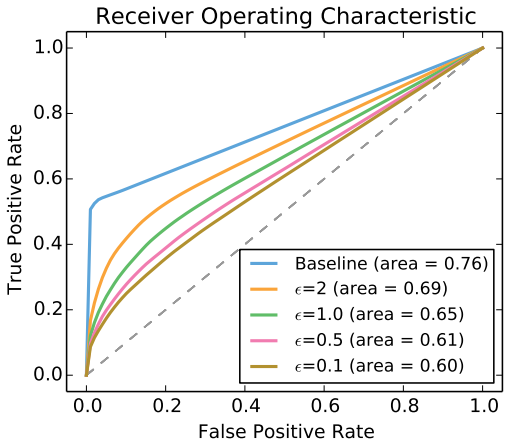

Figure 7.21: User Identification with Speed Obfuscated

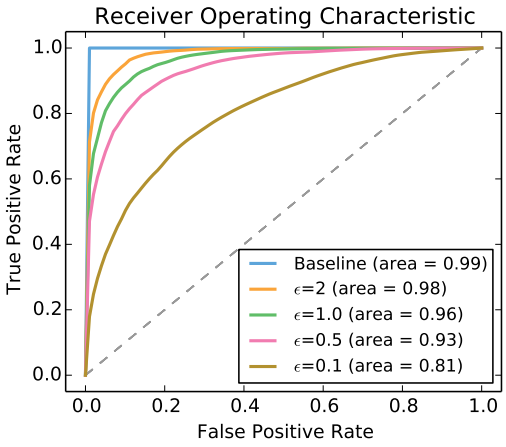

Figure 7.22: Phone Identity Accuracy with Phone Identity Obfuscated

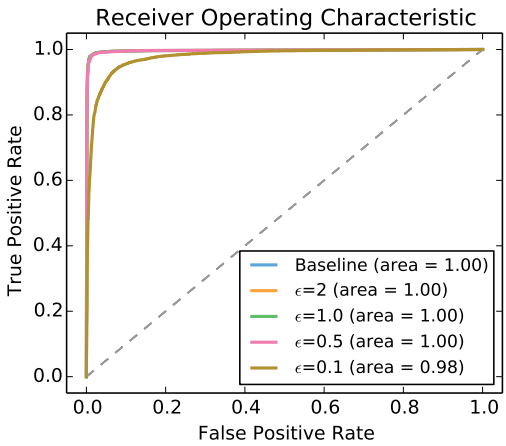

Figure 7.23: Activity Accuracy with Phone Identity Obfuscated

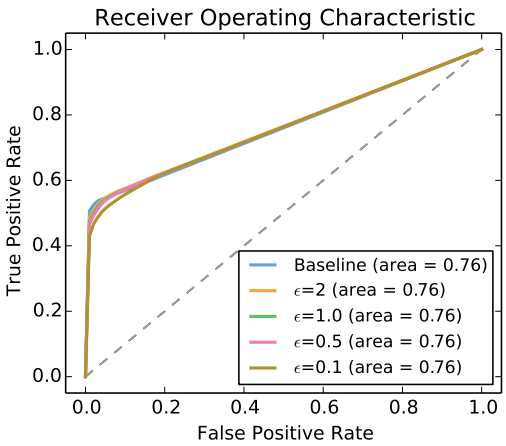

Figure 7.24: User Identification with Phone Identity Obfuscated

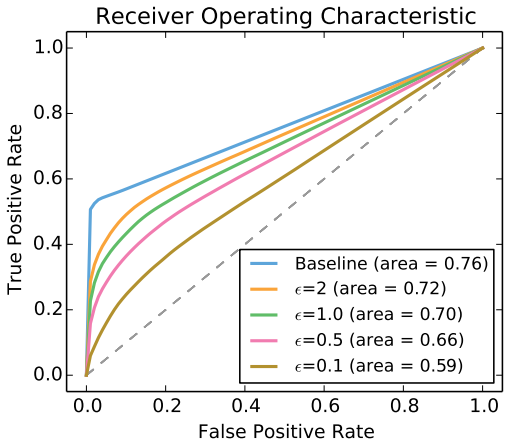

Figure 7.25: User Identification with User Identity Obfuscated

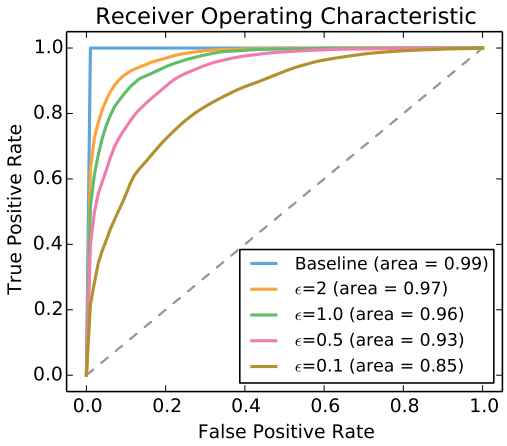

Figure 7.26: Phone Identification with User Identity Obfuscated

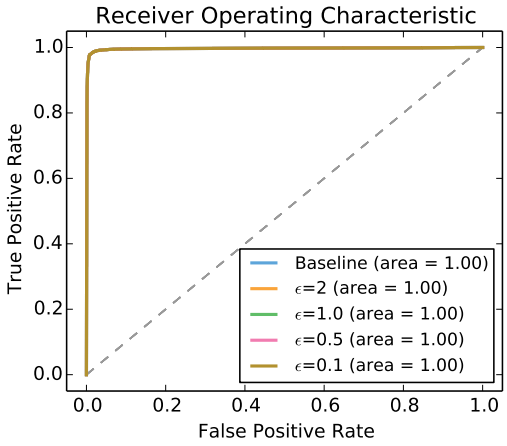

Figure 7.27: Activity Accuracy with User Identity Obfuscated

any obfuscation on speed as seen in Table 7.2.

The analysis in this section have demonstrated the utility of our privacy obfuscation techniques calibrated to noise estimated on the privacy sensitive features. The techniques present promise to hide both user identification and phone identification while inferring walking speed, FEV1/FVC and activity corresponding to success with our first and second scenarios. We see more difficulty when trying to hide health information while maintaining activity. While a negative result for privacy, our results do support the link between activity and health status which is currently being studied in the medical community.

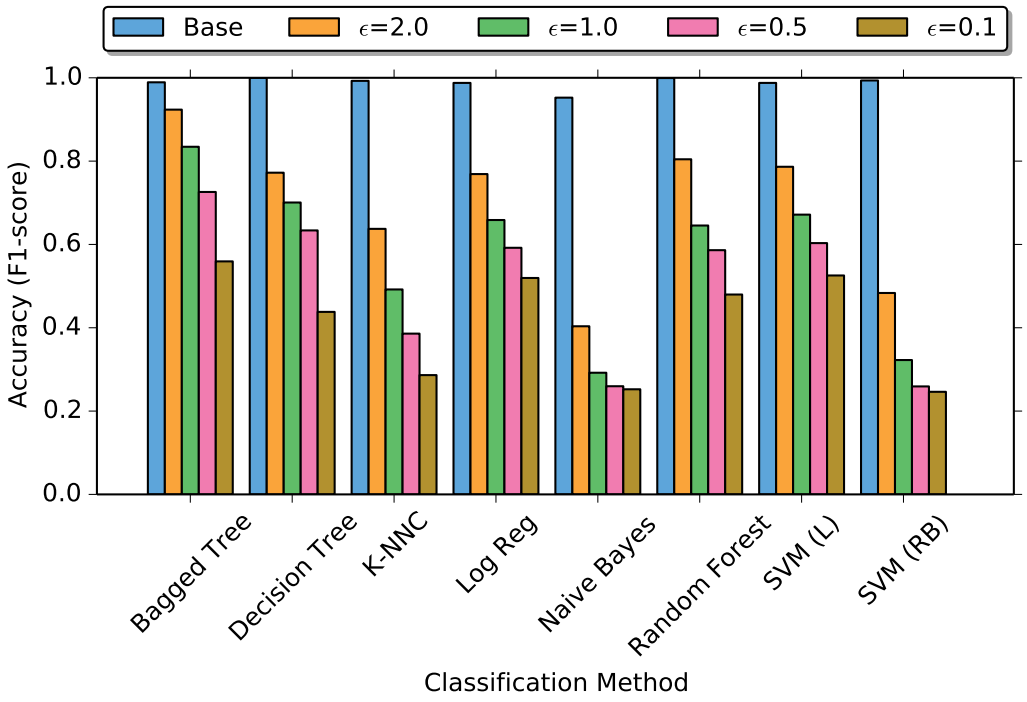

Figure 7.28: Activity Accuracy with Activity Obfuscated over All Models

7.4.3 Obfuscating Features by Model Type

In Section 7.4.2, we demonstrated the ability of our noise models to protect privacy when using the most accurate prediction models. We now repeat the tests using all nine classification and nine regression models to determine if adding noise provides privacy for all model types. We again focus on the activity, user identification, phone identification, FEV1/FVC, and walking speed. We first train the most accurate model by both model type and feature selection as the baseline model. We again introduce speed with various values of ϵ and test the ability of all model types to predict the classifier. To keep the experiments tractable, we only consider the top features from the JMI and SVMFS feature selection methods.

After obfuscation, we generally see decreased prediction performance for all model types with each private target. For example, Figure 7.28 presents the accuracy of predicting the activity after obfuscating the activity. The random forest classifier was originally the most accurate classification model for this prediction. We see that the random forest and decision tree models lose accuracy with lower privacy values more quickly than the bagged tree model. Since the bagged tree model is designed to decrease over fitting

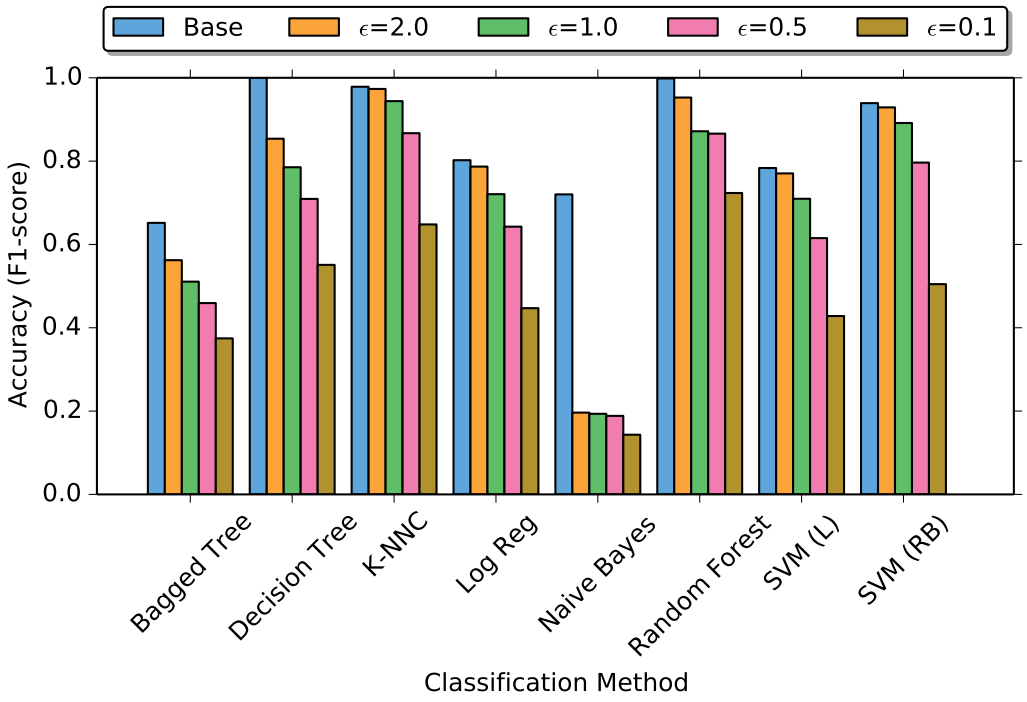

Figure 7.29: Phone Identification Accuracy with Activity Obfuscated over All Models

compared to the decision tree and random forest models, models with less over fitting seem to be less sensitive to our input noise. Intuitively, this is probably due to decision trees having tighter boundaries when incorrectly splitting at leaf nodes due to noise in the training data. Since the bagged tree is more resistant to this training noise and the decision thresholds have a higher range, the bagged tree model is less prone to obfuscation. Overall, we see that all models respond to the added noise with a privacy level of $\epsilon = 1.0$ reducing the maximum $F1$-score from nearly 1.0 to 0.8 and a privacy level of $\epsilon = 0.01$ reducing all models' $F1$-score to a mere 0.5 or random guessing accuracy. Encouragingly, we see the same pattern of decreased prediction accuracy over all private targets tested. Thus, no model tested resists our noise allowing prediction for the private target.

We see the same trends when looking at the ability of various model types to perform predictions on collateral targets. In general, all tests return similar patterns with results either showing the collateral feature can still be predicted with reasonable accuracy or the collateral target is no longer able to be predicted. Figure 7.29 shows an example of a collateral classification

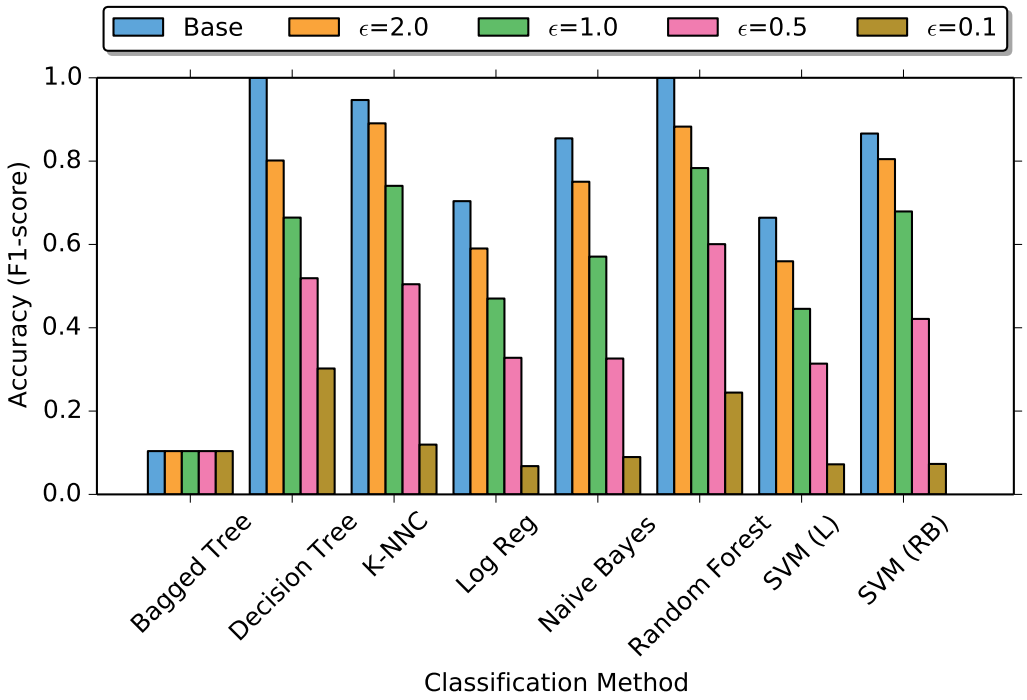

Figure 7.30: User Identification Accuracy with Activity Obfuscated over All Models

which can still be predicted after obfuscation by obfuscating the activity while identifying the phone. Figure 7.30 shows an example of a classification which has decreased accuracy after obfuscating the private target by obfuscating the activity and predicting the user identification. Once again, the best models for prediction have a tendency to be more accurate for prediction after adding noise. We see relatively little decrease in accuracy for the k-NN, random forest, and SVM (RB) for less strict noise thresholds with severe penalties in phone identification once $\epsilon = 0.1$. We notice a less pronounced sensitivity in the random forest classifier. We surmise the model is able to maintain higher accuracy since only the input features which overlap with the private activity features get obfuscation noise. Thus, the model still chooses the correct path in the tree for the non-obfuscated features. The user identification models all see substantial decreases in accuracy when noise is added. The relative drop in accuracy for both phone and user identification seems relative to the initial prediction accuracy of the models without noise.

The effects of adding noise to regression estimates are shown in Figures 7.31 and 7.32. The FEV1/FVC prediction with activity obfuscated demon-

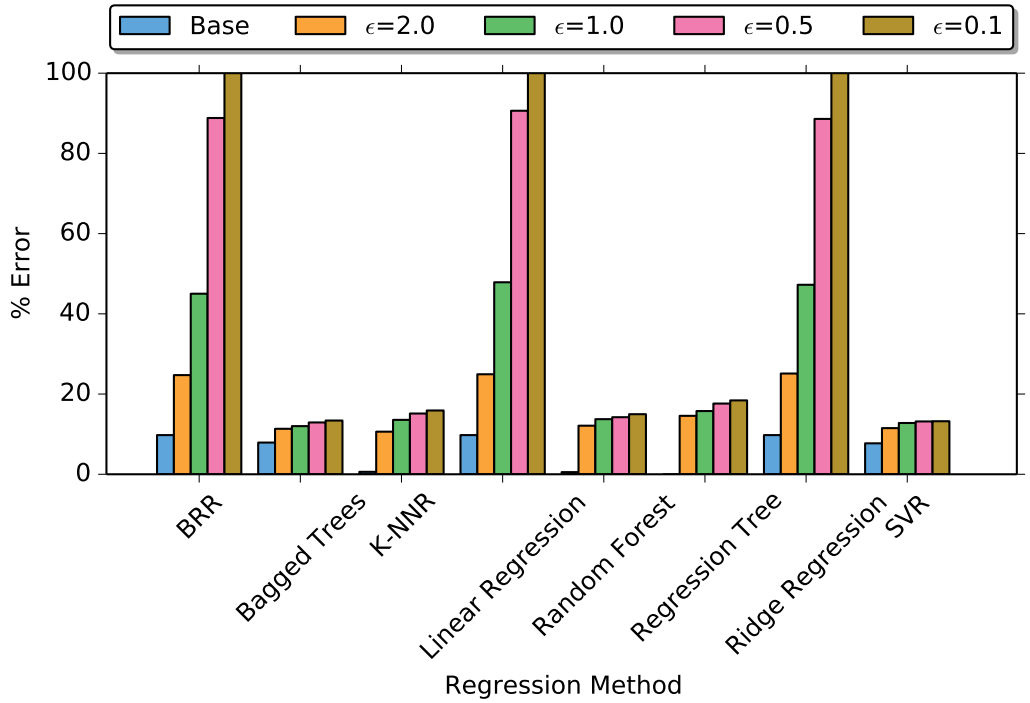

Figure 7.31: FEV1/FVC Accuracy with Activity Obfuscated over All Models

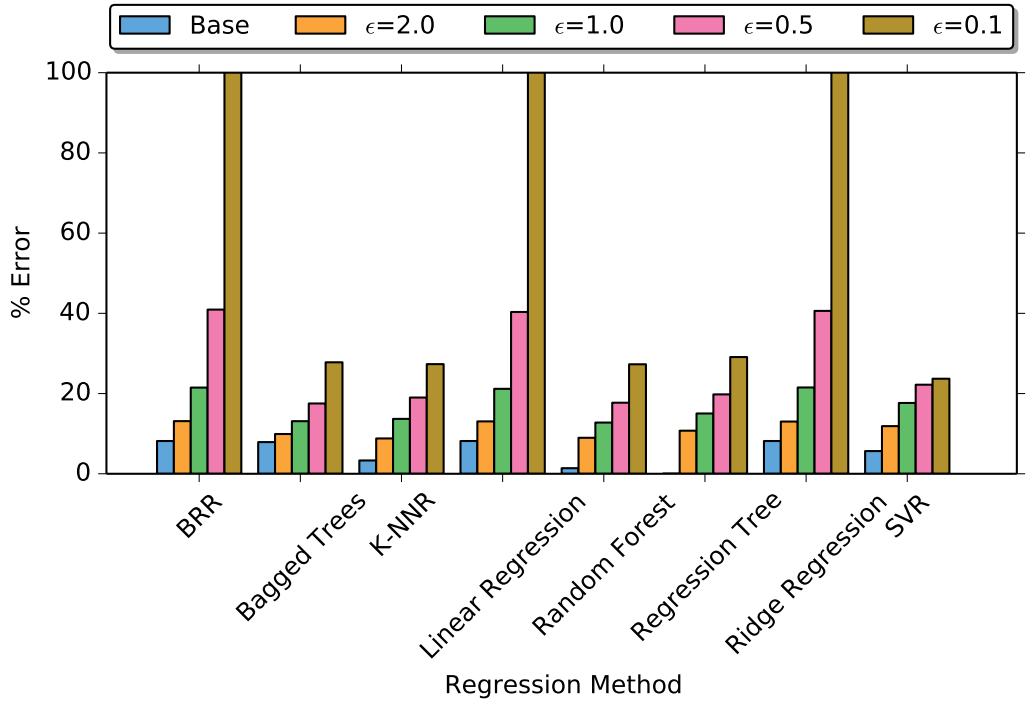

Figure 7.32: Speed Accuracy with Activity Obfuscated over All Models

strates a regression with similar accuracy after adding the noise. The speed prediction with activity obfuscated demonstrates a regression with significant error added after adding the noise. Once again, we see that obfuscating the activity affects both the FEV1/FVC and speed error rates, but certain models for FEV1/FVC resist the noise much better especially the bagged tree and SVR models. We notice that certain regression techniques are prone to massive error rates with the introduction of noise including Bayesian ridge regression, linear regression, and ridge regression. The other models perform with similar accuracy. Once again, both random forest and regression trees perform well without noise, but attain high error rates with even a small amount of obfuscation. All models perform with similar degradation due to noise for the speed estimation. Overall, linear models can produce unrealistic predictions with noise obfuscation while models such as SVR and bagged trees seem most resistant to noise. Both regression trees and random forest produce low errors but lose accuracy quickly with noise indicating a tighter fit within the model itself to the training data.

Over all prediction targets, we see the most accurate models for each prediction tend to continue to be the most accurate after noise is added to the training data. We do see an exception for the random forest, decision tree, and regression tree classifiers since they have a tendency to predict with high accuracy with no noise but quickly lose accuracy when noise is added to the input features of the model. We see that linear regression models often produce values with unrealistically high error rates when tight privacy values are used. Optimistically, we see that all models are affected by obfuscation noise. We also see that collateral damage appears in all predictions we tested; however, the SVM/SVR appears overall to be a good model to assess the impact of obfuscation noise since it maintains high initial prediction accuracy and is less prone to the over fitting of random forest tree models.

7.4.4 Resistance to Further Training

We now test the ability of our noise obfuscation to hide private information even if the attacker trains the machine learning model on noisy data. We do this by repeating the experiment in Section 7.4.3 after retraining all nine classification and regression models using data which is obfuscated using our

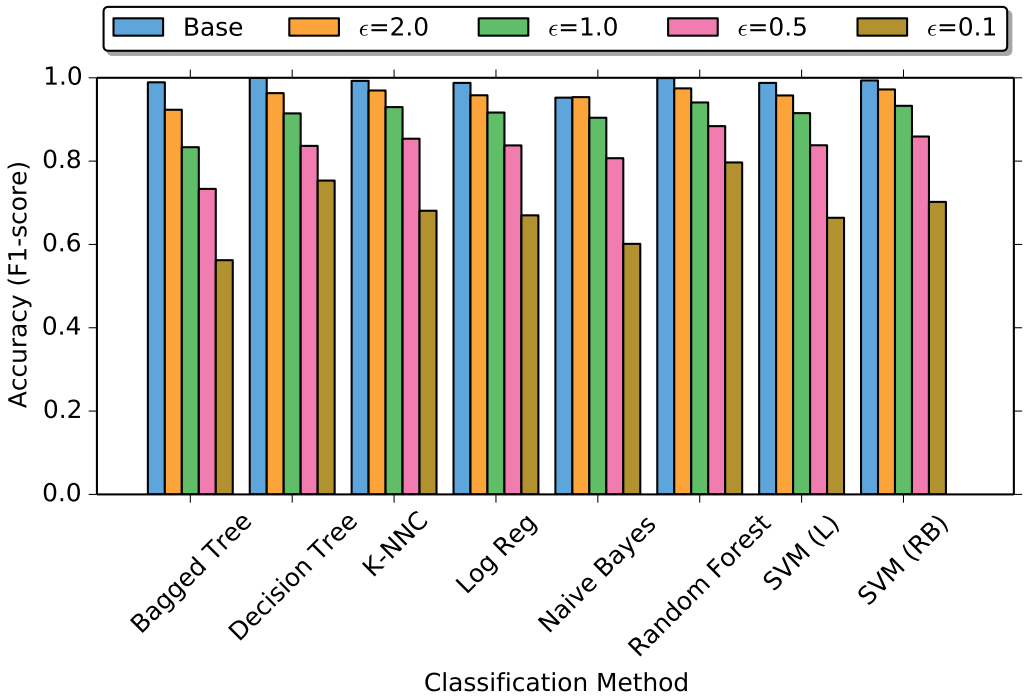

Figure 7.33: Activity Accuracy with Activity Obfuscated over All Models (Noisy Training)

randomly drawn noise. To model realistic conditions, the attacker trains the models on an obfuscated data set. The models are then tested on the same data with the same obfuscation parameter but with the noise resampled. Thus, this models the situation where the attacker trained the models on noisy data and subsequently attempts to predict the private information using samples which have been obfuscated using the same privacy algorithms.

Figure 7.33 illustrates the results of obfuscating the activity and predicting the activity when the models are trained on noisy data. As expected, models trained on data which has been obfuscated with the correct level of noise will predict with higher accuracies then models trained on clean data. However, the added noise is still effective at reducing the prediction accuracy of the activity especially with strict privacy thresholds. For example, we see an average $F1$-score of 0.6 with $\epsilon = 0.1$ over all models. We see the same increased resistance to noise in all tested obfuscations which demonstrate increased accuracy for inference with models trained on noisy data when the noise level is known. We note that knowing the exact level of noise for each feature is the best case scenario for an attacker. We would suggest privacy

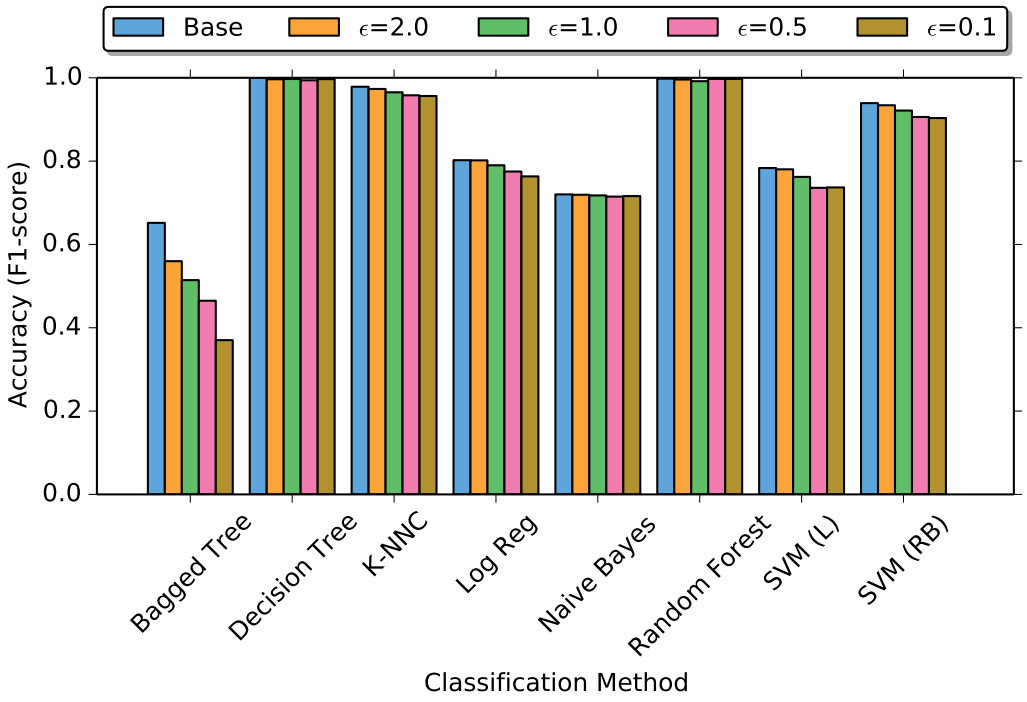

Figure 7.34: Phone Identification Accuracy with Activity Obfuscated over All Models (Noisy Training)

obfuscation mechanisms limit the availability of such information but leave an exploration of varying the noise level randomly to disrupt such improvements in inference accuracy as a topic of future work.

While training on noisy data does slightly decrease the privacy introduced by obfuscation, our testing indicates that it significantly reduces the collateral damage to other predictions. As an example, Figure 7.34 presents the prediction accuracy of the phone when each model is re-trained using noisy data. We see that each model resists noise much better than the models trained on clean data. Recall that only features considered private to activity receive obfuscation. The models trained on noise effectively place more weight on the features which receive no obfuscation. Since these features do not change, the models appear more resistant to obfuscation noise. We see much less resistance for regression models trained on noise. Figure 7.35 shows the output from the speed models after obfuscating activity. We see that training on noisy data eliminates the unrealistic outliers in the linear models; however, all models give similar error rates when introducing noise compared to non-linear models trained without noise.

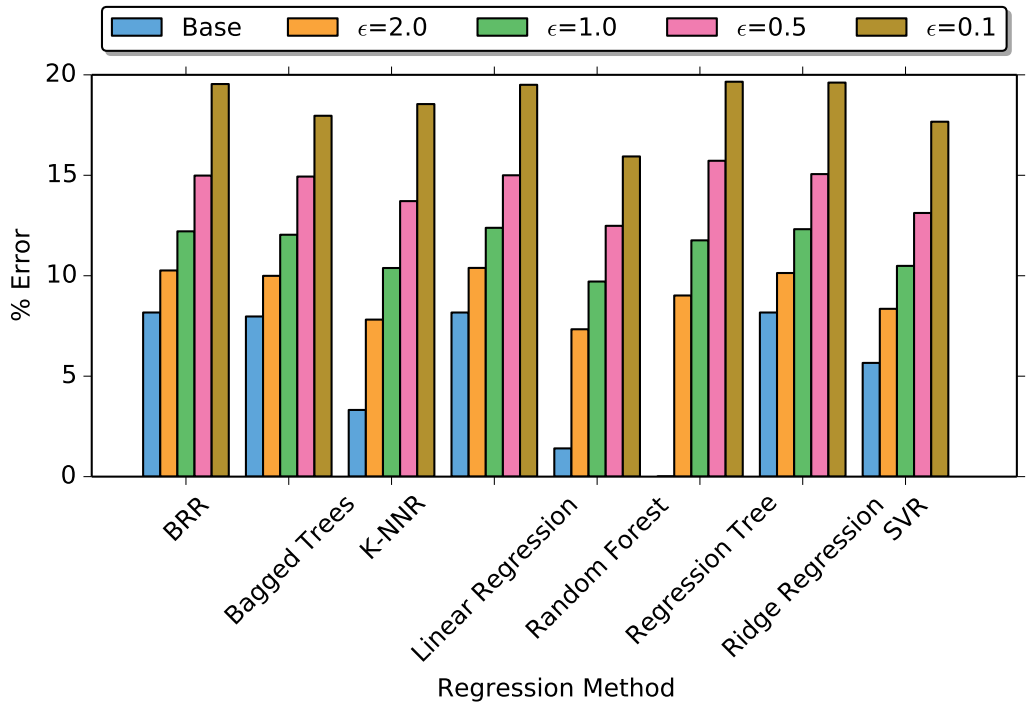

Figure 7.35: Speed Accuracy with Activity Obfuscated over All Models (Noisy Training)

In general, using models trained on noisy data increases the accuracy of making predictions using data with the same level of noise. However even with models trained on noisy data, strict privacy levels still significantly reduce the ability of all model types to make a prediction. The increase in accuracy of models trained with noisy data help to reduce the collateral damage when predicting alternate targets. While all models again perform similarly with obfuscation, we again see tighter fits with random forest and decision trees. We see striking examples where random forest performance significantly decreases such as Figure 7.36 which indicates the random forest model having significantly reduced performance when adding a little noise. Our testing indicates knowing the exact level of noise used to obfuscate the data could be useful for an attacker attempting to infer private information. Conversely, we see knowing the noise could help a legitimate user infer collateral information. Both observations could be useful for the future design of a motion sensor privacy mechanism.

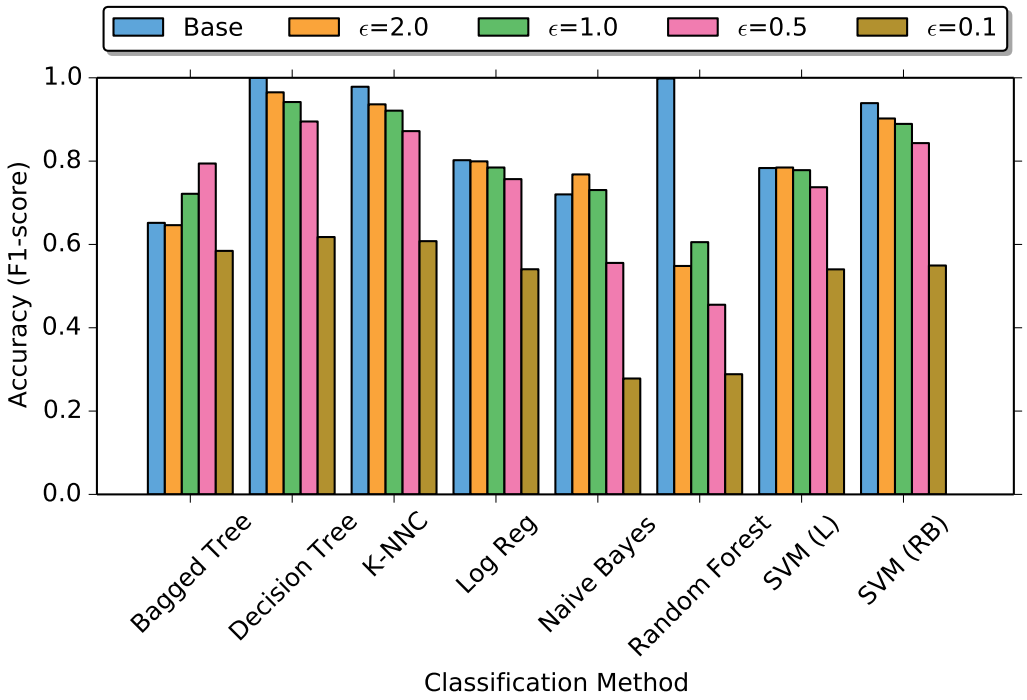

Figure 7.36: Phone Identification Accuracy with Phone Obfuscated over All Models (Noisy Training)

7.5 Conclusions

This chapter used the feature selection methods, optimized models and sensitivity estimates to investigate the ability of hiding specific target features independently of others. As anticipated, significant obfuscation to a single prediction target often causes degradation in the ability to predict other targets. We see three primary tiers of features in our testing. The easiest to obfuscate are the phone and user identification targets which appear to be more sensitive to noise. When we introduce noise according to their sensitivity, we can still predict activity, speed, and FEV1/FVC with reasonable accuracy. Obfuscating activity does not seem to affect phone identification but affects the other classifiers. We finally find that obfuscating speed and FEV1/FVC are challenging without affecting the other targets. We also note that regressions seem to be more sensitive to noise as they return a continuous distribution of values while classification seems to absorb more noise before affecting a change.

We conclude by investigating the ability of each type of model to predict targets when adding obfuscation noise. We find that noise degrades the per-

formance of all model types. Generally, the most accurate model for each prediction type continues to be the most accurate for each noise threshold. Tree-based models including random forest, binary decision, and regression trees which internally contain tight prediction boundaries are more sensitive to over fitting and obfuscation noise. Thus, they see bigger reduction in accuracy for similar noise levels than SVM/SVR models. We finally investigate the effectiveness of noise if the prediction models are trained on obfuscated data. As expected, prediction accuracies increase for all models, but more so for the models predicting alternate targets than the one protected by the obfuscation. Thus, knowing the level of noise could help an attacker gain private information, but would be significantly more helpful to increasing prediction accuracy of collateral targets.

The analysis in this chapter demonstrates the design and testing of privacy obfuscation to hide phone identity while conducting health and fitness monitoring and the ability to hide user identification while conducting fitness and health measurements, two useful scenarios for motion privacy. Our third scenario, predicting fitness while hiding health information, is difficult since our results indicate health and fitness metrics are strongly correlated. We believe our results support the studies indicating fitness metrics could be useful to infer health status and conclude fitness metrics may warrant more careful privacy protection.

CHAPTER 8

CONCLUSIONS

The widespread adoption of mobile devices including fitness devices, medical devices, and smartphones introduce novel threats to users' privacy. We have found mobile devices are collecting continuous traces from motion sensors including accelerometers, gyroscopes and magnetometers. Previous work has demonstrated the usefulness of motion sensors to predict sensitive user information including fitness metrics, user identification and phone identification. This leads to serious privacy concerns since motion sensors are not currently considered sensitive information and are thus fairly easy to access from a malicious phone application. Recently, the use of fitness devices and phones to conduct continuous health monitoring has begun to attract attention. The prospect of monitoring health through a smartphone significantly increases the privacy risk of motion sensors since health information is generally considered high risk data which must be protected. Thus, users must be warned and educated about the information they are leaking by giving apps access to their motion data. Once educated, we expect users will demand systems to provide greater access control to limit the threats to their privacy. However, such access control requires greater analysis and understanding of the sophisticated machine learning techniques used to infer private information from the sensor signals.

To understand the ability of phones to monitor a user's health, we designed and developed software to collect raw motion data with medical grade sensor quality on Android smartphones. We then demonstrated that the motion sensors contained in smartphones could be used to accurately monitor patients and infer health status. With three studies conducted on eighty-eight patients, we presented models to predict unconstrained walking speed useful to conduct the equivalent to a standard six-minute walk test which is used to diagnose chronic obstructive pulmonary disease and congestive heart failure, two serious chronic diseases. We also built models to predict FEV1%,

COPD status and blood oxygen saturation. The trials and predictive models demonstrate the usefulness of using mobile sensors to predict health and provide strong motivation the need for privacy against malicious inference of sensitive information.

The three studies provide a unique data set to both analyze the ability of motion data to be used to predict sensitive information and test the ability of obfuscation to mitigate these threats. The tests collected data from every type of sensor available from our set of ten phones giving thirty-one raw sensor streams including nineteen sensors directly recorded from the phone. Our thirty-one sensors contained the standard set of motion sensors including the accelerometer, gyroscope, orientation, and magnetometer. The remaining twelve sensors were calculated with different reference frames for motion as well as estimation of the user's walking direction and direction of maximum acceleration. For each sensor stream, we use the LibXtract tool kit combined with custom code to extract 74 statistical features giving a total of 2,294 total sensor features in our analysis with roughly 2 GB of raw data.

We demonstrate the ability to predict a comprehensive set of thirty targets including the prediction capability of user identification, phone identification, demographics including age, height, weight and gender. We also looked at activity recognition (walking/non-walking), walking speed, and step counts. Finally, we considered heath status including FEV1%, FEV1/FVC and COPD classification. We split each target classification into periods of walking, non-walking, and a combination of both activities to measure the prediction accuracy both when the user's activity is stationary or unknown and when the user is conducting a known activity such as walking. Our analysis encompasses over forty hours of continuous readings from ten phones and eighty-eight test subjects allowing use to analyze the predictability of our targets.

We explored feature selection for our machine learning models. We evaluated both filter and wrapper methods to predict the top features to train models for each prediction target. We find that filter methods run quickly, but return a different subset of top features than wrapper methods using sequential forward selection with SVM, SVR and k-means models to score features. Understanding the variation in top features is important to understand which features actually need to be obfuscated to protect privacy. The difference in top features returned from feature selection led to the investi-

gation of how to select the set of privacy sensitive features or features which must be hidden in order to diminish the ability of an attacker to infer the private target value. We find that while related to optimal feature selection, selecting the private features is a significantly different problem. We developed three algorithms to identify the private features with the most accurate being a brute force method which conducts a sequential forward feature selection search and classifies the top feature as private per round. Unfortunately, we find that this algorithm takes substantial computational power to run. We found that certain features with high correlation measured by normalized mutual information can be clustered allowing the search to eliminate multiple features per round; however, this led to a high false positive rate. We therefore only clustered features with high normalized mutual information if the features also had similar predictability scores in the first round of the sequential forward top feature search. We found that this method reduced the number of false positives while still significantly reducing runtime.

We evaluated the private feature selection using the subset of features including the magnitude of acceleration, magnitude of gyroscope, and phone orientation. We found that the private feature identification with mutual information and first round SFS scoring completed in roughly half the time of the brute force search while returning fairly low false positive and negatives. We also found that various features had differing numbers of private features with user identification having a relatively low number of private features and activity having a high number of private features. We evaluated with various privacy thresholds for F1-score predictions and mean absolute errors in the sequential forward search identifying the number of private features per privacy threshold. Finally, we evaluated the ability of top feature selection routines to identify the private features finding that the raw normalized mutual information score with the target actually identifies the private features with highest accuracy. We make special note that using the wrapper with k-means clustering to choose private features performed more poorly than other methods considered.

We run the private feature identification with normalized mutual information and first round similarity scoring on the entire data set identifying the private features for all 30 prediction targets. We use the midpoint between the noise threshold or prediction score with random training vectors and the maximum prediction accuracy using an un-optimized SVC or SVR

model. We find that many target predictions contain overlapping features with many correlated features especially between transformations of the motion data. We do see two main clusters of important sensor features with one corresponding to predicting raw motion encompassing data from the accelerometers and another corresponding to rotation being extracted from the orientation and gyroscope sensors. We do eliminate 13 features and 3 sensors which are not useful for prediction from our comprehensive set of sensor features.

We next investigated the ability of machine learning models to predict each target feature and the relative sensitivity of the predictions to changes in the input sensor features. We evaluated nine classification machine learning models and nine regression models with optimized hyperparameters and ten-fold cross validation to accurately measure how well the models can predict the target without significant over fitting. We find that the models themselves do not differ significantly in their ability to predict the target; however, random forest and SVM perform best for classification and SVR performs best for regression. We find that the forward sequential search gives the best input sensor features overall. The filter method giving the highest accuracy is the JMI method. We find that phone and activity identification can be conducted with high accuracy. User identification suffers with ten-fold cross validation due to dropping training targets but performs with leave-one-out validation motivating the need for an individually trained model. We find that COPD status and gender identification have lower accuracy. For regressions, speed, steps, and FEV1/FVC show promise with speed and steps giving the highest accuracy. Age, height, and weight are not predicted with high statistical significance.

We developed strategies to estimate the amount of sensitivity present in sensor features when predicting a target. We develop a library capable of estimating the sensitivity of sklearn machine learning models. We use this to estimate the sensitivity of models trained with both a single input feature to each target and combinations of top features from the feature selection to each target. We compare these sensitivity estimates to the standard sensitivity in differential privacy (the maximum difference between sensor feature values) and confirm that the sensitivity to the decision plane of a classifier and amount of change to the sensor feature per unit change of regression is substantially lower than the worst case differential method. Our methods al-

low us to calibrate noise for obfuscation to lower levels reducing the collateral damage to other prediction targets.

Finally, we tie together the analysis methods presented in the dissertation by taking the top features, the sensitivity estimates and the clinical data set, developing obfuscation strategies, introducing noise, and evaluating the ability to predict the targets while obfuscating specific targets. We compare the sensitivities of five targets including three classification and two regression models. We compare the user identification, phone identification, activity identification, speed prediction, and FEV1/FVC. We first analyze the relative sensitivity of the overlapping features noting that activity has a significant number of overlapping features, speed and FEV1/FVC have low sensitivity and phone and user identification have relatively high sensitivity. We then present obfuscation techniques based on ϵ differential privacy and obfuscate each target using various values for epsilon. We compare the prediction accuracy of each prediction target with the obfuscation yielding twenty-five sets of experiments.

We find that activity, speed, and FEV1/FVC obfuscation significantly diminishes the ability to accurately predict the other targets. Conversely, obfuscating both the user's identity and phone identity can be done with less interference to the other targets. Thus, our framework allows us to design feature level noise capable of hiding individual prediction targets as unobtrusively as possible to protect the usefulness of the signal for other predictions. This allows us to design privacy for two useful scenarios including hiding phone identity during health and fitness sensing and hiding user identity during health and fitness sensing. Our results indicate that health and fitness are closely related motivating the need to have more strict privacy policies with fitness data.

We have presented frameworks to identify private features in motion data and to estimate the sensitivity for each identified private feature. We have also implemented and analyzed a framework to introduce calibrated noise based on differential privacy to obfuscate prediction targets. The next step to this research is to design signal processing techniques to introduce appropriate noise into the sensitive features. For example, we see that the phone identification in particular is sensitive to many features extracted from the rotational rate of the gyroscope. We therefore propose to investigate raw obfuscation into the gyroscope signal in order to affect the gyroscope's fea-

tures. Such analysis will allow the raw signal to be obfuscated in real time and lead the way to the development of privacy frameworks in the firmware which can guarantee privacy.

8.1 Final Thoughts

In this dissertation, we have developed a comprehensive framework to identify, quantify, and analyze private features as well as their sensitivity to prediction. We believe this is a useful first step toward designing privacy frameworks for motion data in real time. The ability to determine private features will allow better policy to protect sensitive features. The framework to determine the sensitivity of the features allows the noise to be carefully calibrated in order to leave a useful signal for other applications. These contributions combined with principles from differential privacy allow sensitive features to be obfuscated with noise producing the least possible damage to unrelated predictions. We hope the work in health prediction in this dissertation can raise awareness to the dangers of releasing raw motion data. We hope the libraries for private feature identification and sensitivity estimation will provide a framework for further analysis to implement privacy frameworks giving users greater control over their private motion data.

We believe the current trend to collect health data from mobile devices is dangerous without access control for a user's data. It is important for users to understand what information both attackers and legitimate companies can obtain from their mobile devices. Furthermore, it will be important for users to be able to protect their personal information without completely blocking access to the motion sensors for every app they may not trust. By understanding the private features necessary to make private predictions and the sensitivity of each feature, we can take the first steps to secure the privacy of users against threats to motion sensor data.

REFERENCES

[1] "Hall effect," 2015. [Online]. Available: http://en.wikipedia.org/wiki/Hall_effect

[2] A. Sandhu, "Information and communications technology: Electronic compass," 2010. [Online]. Available: http://asia.iop.org/cws/article/news/42833

[3] InstrumentationToday, "MEMS accelerometer," 2011. [Online]. Available: http://www.instrumentationtoday.com/mems-accelerometer/2011/08/

[4] J. Esfandyari, "Introduction to MEMS gyroscopes," 2010. [Online]. Available: http://electroiq.com/blog/2010/11/introduction-to-mems-gyroscopes/

[5] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi, "Accelprint: Imperfections of accelerometers make smartphones trackable," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2014.

[6] J. Lee, Y. Kim, and G. J. Welk, "Validity of consumer-based physical activity monitors," *Medicine and Science in Sports and Exercise*, vol. 46, no. 9, pp. 1840–1848, 2014.

[7] J. Takacs, C. L. Pollock, J. R. Guenther, M. Bahar, C. Napier, and M. A. Hunt, "Validation of the Fitbit One activity monitor device during treadmill walking," *Journal of Science and Medicine in Sport*, vol. 17, no. 5, pp. 496–500, 2014.

[8] M. Keally, G. Zhou, G. Xing, J. Wu, and A. Pyles, "PBN: Towards practical activity recognition using smartphone-based body sensor networks," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2011, pp. 246–259.

[9] K. Yatani and K. N. Truong, "Bodyscope: A wearable acoustic sensor for activity recognition," in *Proceedings of the 14th ACM Conference on Ubiquitous Computing (UbiComp)*. ACM, 2012, pp. 341–350.

[10] H. Cao, M. N. Nguyen, C. Phua, S. Krishnaswamy, and X. Li, "An integrated framework for human activity classification," in *Proceedings of the 14th ACM Conference on Ubiquitous Computing (UbiComp)*. ACM, 2012, pp. 331–340.

[11] E. Berlin and K. Van Laerhoven, "Detecting leisure activities with dense motif discovery," in *Proceedings of the 14th ACM Conference on Ubiquitous Computing (UbiComp)*. ACM, 2012, pp. 250–259.

[12] N. D. Lane, Y. Xu, H. Lu, S. Hu, T. Choudhury, A. T. Campbell, and F. Zhao, "Enabling large-scale human activity inference on smartphones using community similarity networks (CSN)," in *Proceedings of the 13th International Conference on Ubiquitous Computing*. ACM, 2011, pp. 355–364.

[13] P. Panuccio, H. Ghasemzadeh, G. Fortino, and R. Jafari, "Power-aware action recognition with optimal sensor selection: An AdaBoost driven distributed template matching approach," in *Proceedings of the First ACM Workshop on Mobile Systems, Applications, and Services for Healthcare*. ACM, 2011, pp. 5:1–5:6.

[14] J. Park, A. Patel, D. Curtis, S. Teller, and J. Ledlie, "Online pose classification and walking speed estimation using handheld devices," in *Proceedings of the 14th ACM Conference on Ubiquitous Computing (UbiComp)*. ACM, 2012, pp. 113–122.

[15] B. G. Steele, L. Holt, B. Belza, S. Ferris, S. Lakshminaryan, and D. M. Buchner, "Quantitating physical activity in COPD using a triaxial accelerometer," *CHEST Journal*, vol. 117, no. 5, pp. 1359–1367, 2000.

[16] R. Moe-Nilssen and J. L. Helbostad, "Estimation of gait cycle characteristics by trunk accelerometry," *Journal of Biomechanics*, vol. 37, no. 1, pp. 121–126, 2004.

[17] F. Pitta, T. Troosters, V. Probst, M. Spruit, M. Decramer, and R. Gosselink, "Quantifying physical activity in daily life with questionnaires and motion sensors in COPD," *European Respiratory Journal*, vol. 27, no. 5, pp. 1040–1055, 2006.

[18] R. A. Rabinovich, Z. Louvaris, Y. Raste, D. Langer, H. Van Remoortel, S. Giavedoni, C. Burtin, E. M. Regueiro, I. Vogiatzis, N. S. Hopkinson, M. I. Polkey, F. J. Wilson, W. MacNee, K. R. Westerterp, and T. Trooster, "Validity of physical activity monitors during daily life in patients with COPD," *European Respiratory Journal*, vol. 42, no. 5, pp. 1205–1215, 2013.

[19] H. Van Remoortel, Y. Raste, Z. Louvaris, S. Giavedoni, C. Burtin, D. Langer, F. Wilson, R. Rabinovich, I. Vogiatzis, N. S. Hopkinson et al., "Validity of six activity monitors in chronic obstructive pulmonary disease: A comparison with indirect calorimetry," *PLoS One*, vol. 7, no. 6, p. e39198, 2012.

[20] F. Pitta, T. Troosters, M. A. Spruit, V. S. Probst, M. Decramer, and R. Gosselink, "Characteristics of physical activities in daily life in chronic obstructive pulmonary disease," *American Journal of Respiratory and Critical Care Medicine*, vol. 171, no. 9, pp. 972–977, 2005.

[21] N. A. Hernandes, D. d. C. Teixeira, V. S. Probst, A. F. Brunetto, E. M. C. Ramos, and F. Pitta, "Profile of the level of physical activity in the daily lives of patients with COPD in Brazil," *Jornal Brasileiro de Pneumologia*, vol. 35, no. 10, pp. 949–956, 2009.

[22] G. K. Pepera, G. R. Sandercock, R. Sloan, J. J. Cleland, L. Ingle, and A. L. Clark, "Influence of step length on 6-minute walk test performance in patients with chronic heart failure," *Physiotherapy*, vol. 98, no. 4, pp. 325–329, 2012.

[23] E. K. Antonsson and R. W. Mann, "The frequency content of gait," *Journal of Biomechanics*, vol. 18, no. 1, pp. 39–47, 1985.

[24] S. A. Skogstad, K. Nymoen, M. E. Høvin, S. Holm, and A. R. Jensenius, "Filtering motion capture data for real-time applications," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 143–147.

[25] D. A. Furtado, A. A. Pereira, A. de Oliveira Andrade, D. P. B. Junior, and M. R. da Silva, "A specialized motion capture system for real-time analysis of mandibular movements using infrared cameras," *Biomedical Engineering Online*, vol. 12, no. 1, p. 17, 2013.

[26] J. Juen, Q. Cheng, and B. Schatz, "Towards a natural walking monitor for pulmonary patients using simple smart phones," in *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*. ACM, 2014, pp. 53–62.

[27] N. Roy, H. Wang, and R. R. Choudhury, "I am a smartphone and I can tell my users walking direction," in *Proceedings of the 12th International Conference on Mobile Systems, Applications, and Services*. ACM, 2014, pp. 329–342.

[28] E. Miluzzo, M. Papandrea, N. D. Lane, H. Lu, and A. T. Campbell, "Pocket, bag, hand, etc. - Automatically detecting phone context through discovery," in *Proceedings of PhoneSense*, 2010, pp. 21–25.

[29] L. Pei, J. Liu, R. Guinness, Y. Chen, H. Kuusniemi, and R. Chen, "Using LS-SVM based motion recognition for smartphone indoor wireless positioning," *Sensors*, vol. 12, no. 5, pp. 6155–6175, 2012.

[30] M. Susi, V. Renaudin, and G. Lachapelle, "Motion mode recognition and step detection algorithms for mobile phone users," *Sensors*, vol. 13, no. 2, pp. 1539–1562, 2013.

[31] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell, "The jigsaw continuous sensing engine for mobile phone applications," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2010, pp. 71–84.

[32] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

[33] H. L. Chu, V. Raman, J. Shen, A. Kansal, V. Bahl, and R. R. Choudhury, "I am a smartphone and I know my user is driving," in *Proceedings of Communication Systems and Networks (COMSNETS)*, 2014, pp. 1–8.

[34] J. Dai, J. Teng, X. Bai, Z. Shen, and D. Xuan, "Mobile phone based drunk driving detection," in *Proceedings of the 4th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*. IEEE, 2010, pp. 1–8.

[35] A. Brajdic and R. Harle, "Walk detection and step counting on unconstrained smartphones," in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2013, pp. 225–234.

[36] A. Mannini and A. M. Sabatini, "On-line classification of human activity and estimation of walk-run speed from acceleration data using support vector machines," in *Proceedings of Engineering in Medicine and Biology Society (EMBS)*. IEEE, 2011, pp. 3302–3305.

[37] H. Vathsangam, A. Emken, D. Spruijt-Metz, and G. S. Sukhatme, "Toward free-living walking speed estimation using Gaussian process-based regression with on-body accelerometers and gyroscopes," in *Proceedings of Pervasive Computing Technologies for Healthcare (PervasiveHealth)*. IEEE, 2010, pp. 1–8.

[38] A. Panagiota, S. Layal, and H. Stefan, "Assessment of human gait speed and energy expenditure using a single triaxial accelerometer," in *Proceedings of the 9th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. IEEE, 2012, pp. 184–188.

[39] H. Vathsangam, A. Tulsyan, and G. Sukhatme, "A data-driven movement model for single cellphone-based indoor positioning," in *Proceedings of the 8th International Conference on Body Sensor Networks (BSN).* IEEE, 2011, pp. 174–179.

[40] Y. Song, S. Shin, S. Kim, D. Lee, and K. H. Lee, "Speed estimation from a tri-axial accelerometer using neural networks," in *Proceedings of the 29th Annual International Conference on Engineering in Medicine and Biology.* IEEE, 2007, pp. 3224–3227.

[41] S. Chen, C. L. Cunningham, J. Lach, and B. C. Bennett, "Extracting spatio-temporal information from inertial body sensor networks for gait speed estimation," in *Proceedings of the 8th International Conference on Wearable and Implantable Body Sensor Networks (BSN).* IEEE, 2011, pp. 71–76.

[42] J. Hu, K. Sun, and C. Cheng, "A kinematic human-walking model for the normal-gait-speed estimation using tri-axial acceleration signals at waist location," *IEEE Transactions on Bio-Medical Engineering*, vol. 60, no. 8, p. 2271, 2013.

[43] M. Yang, H. Zheng, H. Wang, S. McClean, and N. Harris, "Assessing the utility of smart mobile phones in gait pattern analysis," *Health and Technology*, vol. 2, no. 1, pp. 81–88, 2012.

[44] S. Nishiguchi, M. Yamada, K. Nagai, S. Mori, Y. Kajiwara, T. Sonoda, K. Yoshimura, H. Yoshitomi, H. Ito, K. Okamoto et al., "Reliability and validity of gait analysis by android-based smartphone," *Telemedicine and e-Health*, vol. 18, no. 4, pp. 292–296, 2012.

[45] J. Han, E. Owusu, L. T. Nguyen, A. Perrig, and J. Zhang, "Accomplice: Location inference using accelerometers on smartphones," in *Proceedings of the 4th International Conference on Communication Systems and Networks (COMSNETS).* IEEE, 2012, pp. 1–9.

[46] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao, "A reliable and accurate indoor localization method using phone inertial sensors," in *Proceedings of the 14th ACM Conference on Ubiquitous Computing (UbiComp).* ACM, 2012, pp. 421–430.

[47] M. Sousa, A. Techmer, A. Steinhage, C. Lauterbach, and P. Lukowicz, "Human tracking and identification using a sensitive floor and wearable accelerometers," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom).* IEEE, 2013, pp. 166–171.

[48] S. Hilsenbeck, D. Bobkov, G. Schroth, R. Huitl, and E. Steinbach, "Graph-based data fusion of pedometer and wifi measurements for mobile indoor positioning," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2014, pp. 147–158.

[49] A. Kale, A. Rajagopalan, N. Cuntoor, and V. Kruger, "Gait-based recognition of humans using continuous HMMs," in *Proceedings of the 5th IEEE International Conference on Automatic Face and Gesture Recognition*. IEEE, 2002, pp. 336–341.

[50] M. S. Nixon and J. N. Carter, "Automatic recognition by gait," *Proceedings of the IEEE*, vol. 94, no. 11, pp. 2013–2024, 2006.

[51] J. E. Boyd and J. J. Little, "Biometric gait recognition," in *Advanced Studies in Biometrics*. Springer, 2005, pp. 19–42.

[52] D. Gafurov, K. Helkala, and T. Søndrol, "Gait recognition using acceleration from MEMS," in *Proceedings of the 1st International Conference on Availability, Reliability and Security*. IEEE, 2006, pp. 432–437.

[53] M. Tanviruzzaman and S. I. Ahamed, "Your phone knows you: Almost transparent authentication for smartphones," in *Proceedings of the 38th Annual Computer Software and Applications Conference (COMPSAC)*. IEEE, 2014, pp. 374–383.

[54] J. E. Bardram, R. E. Kjær, and M. Ø. Pedersen, "Context-aware user authentication–Supporting proximity-based login in pervasive computing," in *Proceedings of the 5th International Conference on Ubiquitous Computing (UbiComp)*. Springer, 2003, pp. 107–123.

[55] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Cell phone-based biometric identification," in *Proceedings of the 4th International Conference on Biometrics: Theory Applications and Systems (BTAS)*. IEEE, 2010, pp. 1–7.

[56] J. Lester, C. Hartung, L. Pina, R. Libby, G. Borriello, and G. Duncan, "Validated caloric expenditure estimation using a single body-worn sensor," in *Proceedings of the 11th International Conference on Ubiquitous Computing*. ACM, 2009, pp. 225–234.

[57] A. Zhan, M. Chang, Y. Chen, and A. Terzis, "Accurate caloric expenditure of bicyclists using cellphones," in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*. ACM, 2012, pp. 71–84.

[58] E. Ertin, N. Stohs, S. Kumar, A. Raij, M. al'Absi, and S. Shah, "Autosense: Unobtrusively wearable sensor suite for inferring the onset, causality, and consequences of stress in the field," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2011, pp. 274–287.

[59] H. Lu, D. Frauendorfer, M. Rabbi, M. S. Mast, G. T. Chittaranjan, A. T. Campbell, D. Gatica-Perez, and T. Choudhury, "Stresssense: Detecting stress in unconstrained acoustic environments using smartphones," in *Proceedings of the 14th International Conference on Ubiquitous Computing*. ACM, 2012, pp. 351–360.

[60] E. C. Larson, M. Goel, G. Boriello, S. Heltshe, M. Rosenfeld, and S. N. Patel, "Spirosmart: Using a microphone to measure lung function on a mobile phone," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 2012, pp. 280–289.

[61] M. Rabbi, S. Ali, T. Choudhury, and E. Berke, "Passive and in-situ assessment of mental and physical well-being using mobile sensors," in *Proceedings of the 13th International Conference on Ubiquitous Computing*, 2011, pp. 385–394.

[62] G. Yavuz, M. Kocak, G. Ergun, H. Alemdar, H. Yalcin, O. D. Incel, and C. Ersoy, "A smartphone based fall detector with online location support," in *Proceedings of the International Workshop on Sensing for App Phones; Zurich, Switzerland*, 2010, pp. 31–35.

[63] Chipworks, "Chipworks.com," April 2013. [Online]. Available: http://www.chipworks.com/

[64] A. Albarbar, A. Badri, J. K. Sinha, and A. Starr, "Performance evaluation of MEMS accelerometers," *Measurement*, vol. 42, no. 5, pp. 790–795, 2009.

[65] R. LeMoyne, C. Coroian, T. Mastroianni, and W. Grundfest, "Accelerometers for quantification of gait and movement disorders: A perspective review," *Journal of Mechanics in Medicine and Biology*, vol. 8, no. 02, pp. 137–152, 2008.

[66] D. John and P. Freedson, "Actigraph and Actical physical activity monitors: A peek under the hood," *Medicine and Science in Sports and Exercise*, vol. 44, pp. S86–S89, 2012.

[67] D. Brooks and S. Solway, "ATS statement on six-minute walk test," *American Journal of Respiratory and Critical Care Medicine*, vol. 167, no. 9, pp. 1287–1287, 2003.

[68] Q. Cheng, J. Juen, Y. Li, V. Prieto-Centurion, J. A. Krishnan, and B. R. Schatz, "Gaittrack: Health monitoring of body motion from spatio-temporal parameters of simple smart phones," in *Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics (BCB)*. ACM, 2013, pp. 897–906.

[69] J. Juen, Q. Cheng, V. Prieto-Centurion, J. A. Krishnan, and B. Schatz, "Health monitors for chronic disease by gait analysis with mobile phones," *Telemedicine and e-Health*, pp. 1035–1041, 2014.

[70] J. Juen, Q. Cheng, and B. Schatz, "A natural walking monitor for pulmonary patients using mobile phones," *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 4, pp. 1399–1405, 2015.

[71] Q. Cheng, J. Juen, and B. R. Schatz, "Using mobile phones to simulate pulse oximeters: Gait analysis predicts oxygen saturation," in *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*. ACM, 2014, pp. 331–340.

[72] Q. Cheng, J. Juen, J. Hsu-Lumetta, and B. Schatz, "Predicting transitions in oxygen saturation using phone sensors," *Telemedicine and e-Health*.

[73] J. Bullock and U. Conservatoire, "Libxtract: A lightweight library for audio feature extraction," in *Proceedings of the International Computer Music Conference*, vol. 43, 2007.

[74] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[75] A. Pocock and G. Brown, "Feast," 2014. [Online]. Available: http://mloss.org/software/view/386/

[76] G. Brown, A. Pocock, M.-J. Zhao, and M. Luján, "Conditional likelihood maximisation: A unifying framework for information theoretic feature selection," *The Journal of Machine Learning Research*, vol. 13, pp. 27–66, 2012.

[77] K. Torkkola, "Feature extraction by non parametric mutual information maximization," *The Journal of Machine Learning Research*, vol. 3, pp. 1415–1438, 2003.

[78] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Transactions on Neural Networks*, vol. 20, no. 2, pp. 189–201, 2009.

[79] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[80] J. Novovičová, P. Somol, M. Haindl, and P. Pudil, "Conditional mutual information based feature selection for classification task," in *Progress in Pattern Recognition, Image Analysis and Applications*. Springer, 2007, pp. 417–426.

[81] H. Yang and J. Moody, "Feature selection based on joint mutual information," in *Proceedings of the International ICSC Symposium on Advances in Intelligent Data Analysis*. Citeseer, 1999, pp. 22–25.

[82] F. Fleuret, "Fast binary feature selection with conditional mutual information," *The Journal of Machine Learning Research*, vol. 5, pp. 1531–1555, 2004.

[83] S. Ullman, E. Sali, and M. Vidal-Naquet, "A fragment-based approach to object representation and classification," in *Visual Form 2001*. Springer, 2001, pp. 85–100.

[84] A. Jakulin, "Machine learning based on attribute interactions," Ph.D. dissertation, Univerza v Ljubljani, 2005.

[85] P. E. Meyer, C. Schretter, and G. Bontempi, "Information-theoretic feature selection in microarray data using variable complementarity," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 3, pp. 261–274, 2008.

[86] J. Kittler, "Feature selection and extraction," *Handbook of Pattern Recognition and Image Processing*, pp. 59–83, 1986.

[87] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[88] T. Griffiths, "Gibbs sampling in the generative model of latent Dirichlet allocation," Stanford University, Tech. Rep.

[89] G. Casella and E. I. George, "Explaining the Gibbs sampler," *The American Statistician*, vol. 46, no. 3, pp. 167–174, 1992.

[90] A. Riddell, "Topic modeling with latent Dirichlet allocation," 2015. [Online]. Available: https://github.com/ariddell/lda/

[91] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21–27, 1967.

[92] T. F. Chan, G. H. Golub, and R. J. LeVeque, "Updating formulae and a pairwise algorithm for computing sample variances," in *Proceedings of COMPSTAT 1982 5th Symposium Held at Toulouse 1982*. Springer, 1982, pp. 30–41.

[93] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Transactions on Electronic Computers*, no. 3, pp. 326–334, 1965.

[94] C.-Y. Lee, "Representation of switching circuits by binary-decision programs," *Bell System Technical Journal*, vol. 38, no. 4, pp. 985–999, 1959.

[95] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. CRC Press, 1984.

[96] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[97] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[98] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[99] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[100] D. J. MacKay, "Bayesian interpolation," *Neural Computation*, vol. 4, no. 3, pp. 415–447, 1992.

[101] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[102] C. E. Rasmussen, "Gaussian processes for machine learning," in *Proceedings of Adaptive Computation and Machine Learning*. Citeseer, 2006.

[103] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg et al., "Scikit-learn: Machine learning in Python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[104] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography*. Springer, 2006, pp. 265–284.

[105] D. Kifer and A. Machanavajjhala, "A rigorous and customizable framework for privacy," in *Proceedings of the 31st Symposium on Principles of Database Systems*. ACM, 2012, pp. 77–88.

[106] P. Mohan, A. Thakurta, E. Shi, D. Song, and D. Culler, "GUPT: Privacy preserving data analysis made easy," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, 2012, pp. 349–360.

[107] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. ACM, 2013, pp. 901–914.

[108] K. Chatzikokolakis, C. Palamidessi, and M. Stronati, "A predictive differentially-private mechanism for mobility traces," in *Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science, E. De Cristofaro and S. Murdoch, Eds. Springer International Publishing, 2014, vol. 8555, pp. 21–41. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-08506-7_2